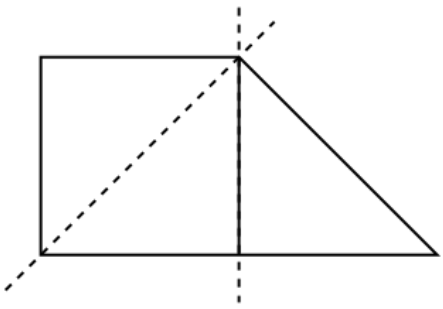
Информатика. 2 класс
Решения

1 вариант

1. (10 баллов) Пять человек с роялем хотят переправиться на лодке на другой берег реки. Лодка вмещает либо двух человек и рояль, либо трёх человек. Рояль очень тяжелый, и поднять его может только 5 человек сразу. Смогут ли они перебраться?
   Решение:
   Да, сначала все пять человек ставят рояль в лодку, и 2 человека с роялем переплывают реку, один остается на дальнем берегу, второй возвращается за третьим, оставляет его на дальнем берегу, затем за четвёртым, затем за пятым, и они вместе вытаскивают рояль из лодки.

2. (25 баллов) За круглым столом сидят 9 жителей Острова Фей и Гномов. Гномы всегда лгут, а феи говорят только правду. Каждый из них сказал: «Мои соседи – фея и гном». Сколько среди них гномов?
   Решение:
   3 или 9. Рядом с каждой феей сидит гном, поэтому гномы есть. Соседями гнома могут оказаться либо два гнома, либо две феи. Если у гнома оба соседа гномы, то и дальше по кругу нет ни одной феи. Если оба соседа лжеца феи, то за каждой феей сидит ещё фея, затем гном, затем две феи, и т.д. Гномов трое.
3. (15 баллов) У Пети и Васи по одинаковому бумажному многоугольнику. Каждый из них перегнул по прямой и обвёл получившуюся фигуру. У Васи получился квадрат. Мог ли у Пети получиться треугольник с острыми углами?
   Решение:
   Да, например



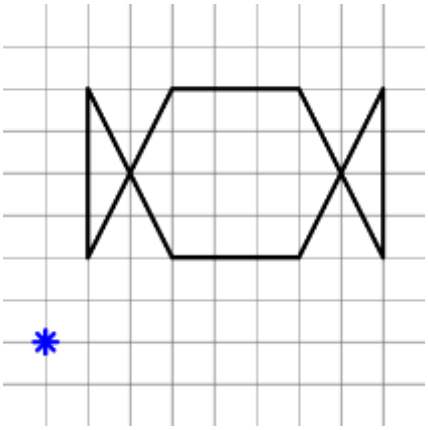
4. (30 баллов) Робот-чертежник умеет, двигаясь по клеточкам, оставлять за собой след в виде линии с помощью пера. У робота имеется следующий набор команд:
   - **опустить перо;**
   - **поднять перо;**

- **сместиться на(x, y).**

Команда **сместиться на(x, y)** перемещает робота на **x** вправо или на **x** влево, если число отрицательное, и на **y** вверх или вниз.

Например, если в начале работы программы робот стоит в точке обозначенной звездочкой, то в результате выполнения программы:

1. **сместиться на(1, 2)**
2. **опустить перо**
3. **сместиться на(0, 4)**
4. **сместиться на(2, -4)**
5. **сместиться на(3, 0)**
6. **сместиться на(2, 4)**
7. **сместиться на(0, 4)**
8. **сместиться на(-2, 4)**
9. **сместиться на(-3, 0)**
10. **сместиться на(-2, -4)**
11. **поднять перо**

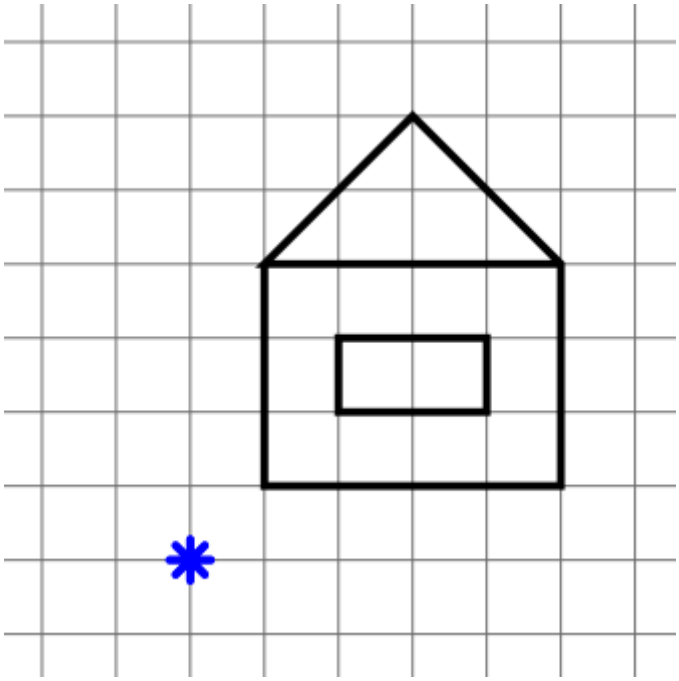


Получится такой рисунок (справа):

Нарисуйте след, который оставит робот, выполнив следующую программу:

1. **сместиться на(1, 4)**
2. **опустить перо**
3. **сместиться на(0, -3)**
4. **сместиться на(4, 0)**
5. **сместиться на(0, 3)**
6. **сместиться на(-2, 2)**
7. **сместиться на(-2, -2)**
8. **сместиться на(4, 0)**
9. **поднять перо**
10. **сместиться на(-1, -1)**
11. **опустить перо**
12. **сместиться на(-2, 0)**
13. **сместиться на(0, -1)**
14. **сместиться на(2, 0)**
15. **сместиться на(1,0)**
16. **поднять перо**
17. **сместиться на(-4, 3)**

Решение:



5. (20 баллов) Коля решил зашифровать свой пароль, состоящий только из русских букв А, Б, Е, К, Л, О. Каждая буква кодировалась двоичным словом по таблице:

А	**Б**	**Е**	**К**	**Л**	**О**
10	11	101	001	00	011

В итоге он получил следующую запись:

011 11 101 00 10 011

011 11 101 001 00 11

Расшифруйте и узнайте, какой пароль использовал Коля.
Решение: обелао, ОБЕКЛБ

Информатика. 2 класс
Решения

2 вариант

1. (10 баллов) Шесть человек с роялем хотят переправиться на лодке на другой берег реки. Лодка вмещает либо двух человек и рояль, либо трёх человек. Рояль очень тяжелый, и поднять его может только 6 человек сразу. Смогут ли они перебраться?

   Решение:

Да, сначала все шесть человек ставят рояль в лодку, и 2 человека с роялем переплывают реку, один остается на дальнем берегу, второй возвращается за третьим, оставляет его на дальнем берегу, затем за четвёртым, и т.д., и они вместе вытаскивают рояль из лодки.

2. (25 баллов) За круглым столом сидят 12 жителей Острова Фей и Гномов. Гномы всегда лгут, а феи говорят только правду. Каждый из них сказал: «Мои соседи – фея и гном». Сколько среди них гномов?

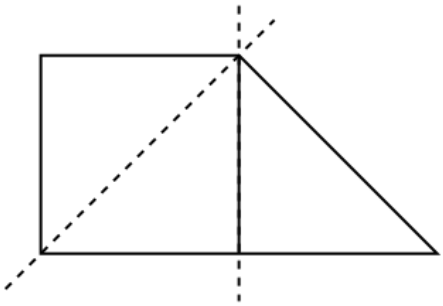
   Решение:

4 или 12. Рядом с каждой феей сидит гном, поэтому гномы есть. Соседями гнома могут оказаться либо два гнома, либо две феи. Если у гнома оба соседа гномы, то и дальше по кругу нет ни одной феи. Если оба соседа гнома феи, то за каждой феей сидит ещё фея, затем гном, затем две феи, и т.д. Гномов четверо.

3. (15 баллов) У Маши и Кати по одинаковому бумажному многоугольнику. Каждая из них перегнула по прямой и обвела получившуюся фигуру. У Маши получился квадрат. Мог ли у Кати получиться треугольник?

   Решение:

   Да, например



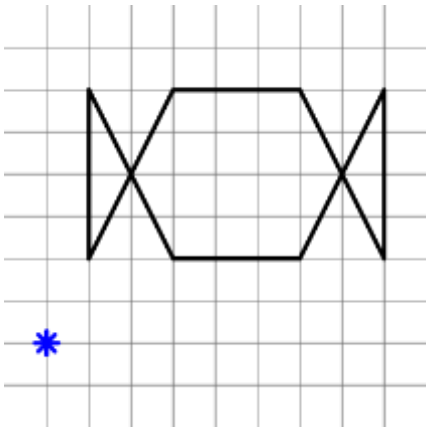
4. (30 баллов) Робот-чертежник умеет, двигаясь по клеточкам, оставлять за собой след в виде линии с помощью пера. У робота имеется следующий набор команд:
   - **опустить перо;**

- **поднять перо;**
- **сместиться на(x, y).**

Команда **сместиться на(x, y)** перемещает робота на **x** вправо или на **x** влево, если число отрицательное, и на **y** вверх или вниз.

Например, если в начале работы программы робот стоит в точке обозначенной звездочкой, то в результате выполнения программы:

1. **сместиться на(1, 2)**
2. **опустить перо**
3. **сместиться на(0, 4)**
4. **сместиться на(2, -4)**
5. **сместиться на(3, 0)**
6. **сместиться на(2, 4)**
7. **сместиться на(0, 4)**
8. **сместиться на(-2, 4)**
9. **сместиться на(-3, 0)**
10. **сместиться на(-2, -4)**
11. **поднять перо**

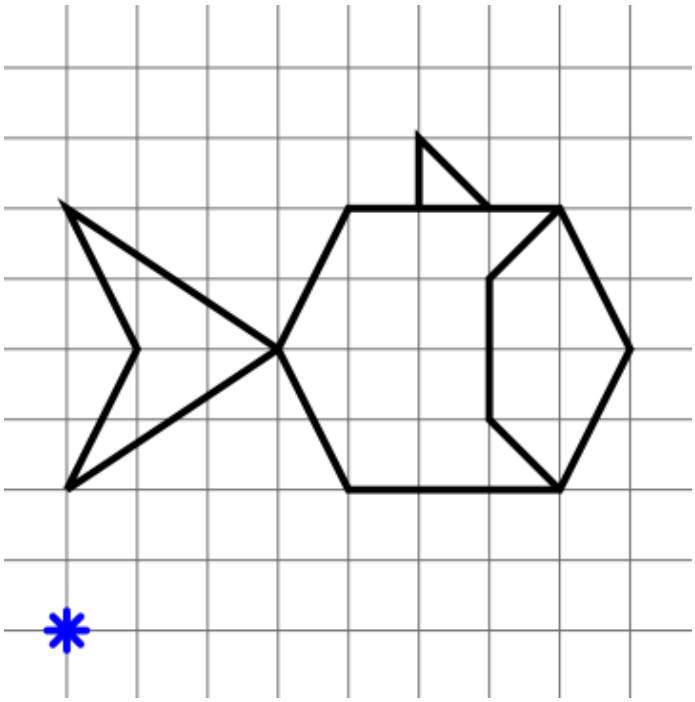


Получится такой рисунок (справа):

Нарисуйте след, который оставит робот, выполнив следующую программу:

1. **сместиться на(0, 2)**
2. **опустить перо**
3. **сместиться на(1, 2)**
4. **сместиться на(-1, 2)**
5. **сместиться на(3, -2)**
6. **сместиться на(1, 2)**
7. **сместиться на(2, 0)**
8. **сместиться на(-1, 1)**
9. **сместиться на(0, -1)**
10. **сместиться на(0, 2)**
11. **сместиться на(1, -2)**
12. **сместиться на(-1, -2)**
13. **сместиться на(-3, 0)**
14. **сместиться на(-1, 2)**
15. **сместиться на(-3, -2)**
16. **поднять перо**
17. **сместиться на(7, 0)**
18. **опустить перо**
19. **сместиться на(-1, 1)**
20. **сместиться на(0, 2)**
21. **сместиться на(1, 1)**
22. **поднять перо**

Решение:



5. (20 баллов) Коля решил зашифровать свой пароль, состоящий только из русских букв А, Б, Е, К, Л, О. Каждая буква кодировалась двоичным словом по таблице:

А	**Б**	**Е**	**К**	**Л**	**О**
10	11	101	001	00	011

В итоге он получил следующую запись:

00 011 001 10 11 101 00

00 011 00 11 011 101 00

Расшифруйте и узнайте, какой пароль использовал Коля.
Решение: локабел, лолбоел

Информатика. 2 класс
Решения

3 вариант

1. (10 баллов) Семь человек с роялем хотят переправиться на лодке на другой берег реки. Лодка вмещает либо двух человек и рояль, либо трёх человек. Рояль очень тяжелый, и поднять его может только 7 человек сразу. Смогут ли они перебраться?

   Решение:

Да, сначала все семь человек ставят рояль в лодку, и 2 человека с роялем переплывают реку, один остается на дальнем берегу, второй возвращается за третьим, оставляет его на дальнем берегу, затем за четвёртым, и т.д., и они вместе вытаскивают рояль из лодки.

2. (25 баллов) За круглым столом сидят 15 жителей Острова Фей и Гномов. Гномы всегда лгут, а феи говорят только правду. Каждый из них сказал: «Мои соседи – фея и гном». Сколько среди них фей?

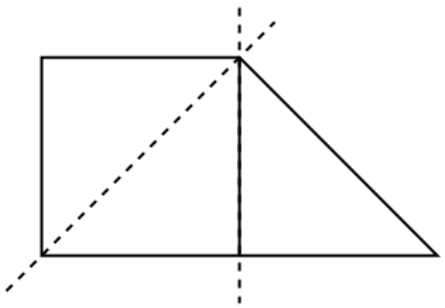
   Решение:

   10 или 0. Рядом с каждой феей сидит гном, поэтому гномы есть. Соседями гнома могут оказаться либо два гнома, либо две феи. Если у гнома оба соседа гномы, то и дальше по кругу нет ни одной феи. Если оба соседа гнома феи, то за каждой феей сидит ещё фея, затем гном, затем две феи, и т.д. Гномов пятеро, значит, фей 10.

3. (15 баллов) У Маши и Кати по одинаковому бумажному многоугольнику. Каждая из них перегнула по прямой и обвела получившуюся фигуру. У Маши получился квадрат. Мог ли у Кати получиться треугольник?

   Решение:

   Да, например



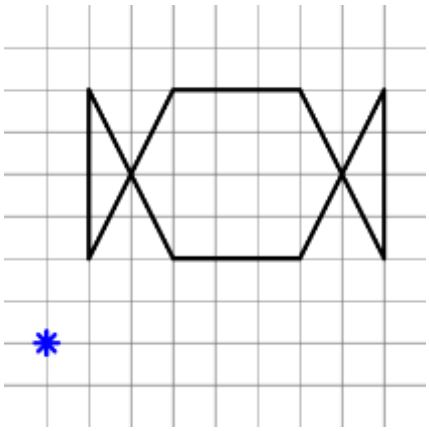
4. (30 баллов) Робот-чертежник умеет, двигаясь по клеточкам, оставлять за собой след в виде линии с помощью пера. У робота имеется следующий набор команд:

- **опустить перо;**
- **поднять перо;**
- **сместиться на(x, y).**

Команда **сместиться на(x, y)** перемещает робота на **x** вправо или на **x** влево, если число **x** отрицательное, и на **y** вверх или вниз.

Например, если в начале работы программы робот стоит в точке обозначенной звездочкой, то в результате выполнения программы:

1. **сместиться на(1, 2)**
2. **опустить перо**
3. **сместиться на(0, 4)**
4. **сместиться на(2, -4)**
5. **сместиться на(3, 0)**
6. **сместиться на(2, 4)**
7. **сместиться на(0, 4)**
8. **сместиться на(-2, 4)**
9. **сместиться на(-3, 0)**
10. **сместиться на(-2, -4)**
11. **поднять перо**

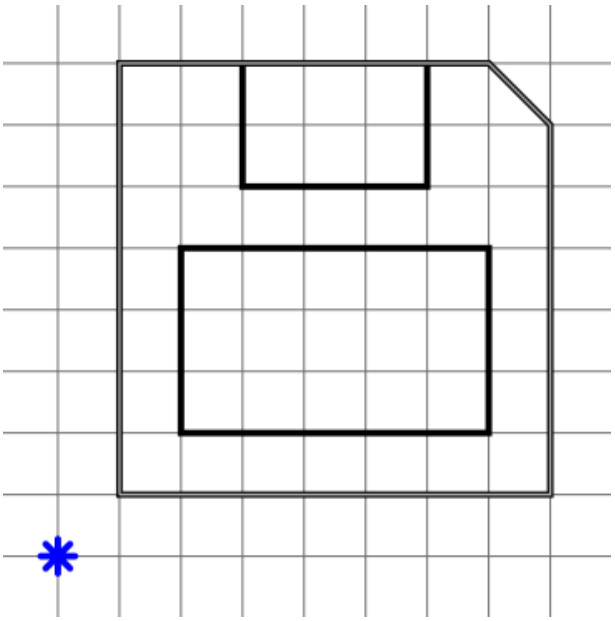


Получится такой рисунок (справа):

Нарисуйте след, который оставит робот, выполнив следующую программу:

1. **сместиться на(1, 1)**
2. **опустить перо**
3. **сместиться на(0, 7)**
4. **сместиться на(2, 0)**
5. **сместиться на(0, -2)**
6. **сместиться на(3, 0)**
7. **сместиться на(0, 2)**
8. **сместиться на(1, 0)**
9. **сместиться на(1, -1)**
10. **сместиться на(0, 6)**
11. **сместиться на(-7, 0)**
12. **поднять перо**
13. **сместиться на(1, 1)**
14. **опустить перо**
15. **сместиться на(5,0)**
16. **сместиться на(3, 0)**
17. **сместиться на(-5, 0)**
18. **сместиться на(-3, 0)**
19. **поднять перо**
20. **сместиться на(1, 6)**
21. **опустить перо**
22. **сместиться на(3, 0)**

Решение:



5. (20 баллов) Коля решил зашифровать свой пароль, состоящий только из русских букв А, Б, Е, К, Л, О. Каждая буква кодировалась двоичным словом по таблице:

А	**Б**	**Е**	**К**	**Л**	**О**
10	11	101	001	00	011

В итоге он получил следующую запись:

011 11 101 001 10

Расшифруйте и узнайте, какой пароль использовал Коля.
Решение: обелка

Информатика. 2 класс
Критерии оценивания

1. Обоснованно получен верный ответ – 10 баллов
   Приведено в целом верное решение, но имеются неточности – 5 баллов
   Дан верный ответ “да” без обоснования – 1 балл
   Решение не удовлетворяет ни одному критерию выше, однако в рассуждениях имеются продвижения – 1 балл
   Любой другой ответ – 0 баллов

2. Обоснованно получен верный ответ – 25 баллов
   Дан верный числовой ответ без обоснования – +2 балла за каждый
   Дано верное обоснование одного верного числового ответа – +10 баллов
   Дано в целом верное обоснование одного числового ответа, но имеются неточности – +5 баллов
   Решение не удовлетворяет ни одному критерию выше, однако в рассуждениях имеются продвижения – 5 баллов
   Любой другой ответ – 0 баллов

3. Обоснованно получен верный ответ, приведен пример многоугольника, удовлетворяющего условию задачи, показаны прямые, по которым его складывали и понятен результат складывания – 15 баллов
   Решение в целом верно, почти удовлетворяет критерию выше, однако имеются неточности – 10 баллов
   Решение не удовлетворяет ни одному критерию выше, однако в рассуждениях имеются продвижения – 3 балла
   Дан верный ответ “да” без обоснования – 1 балл
   Любой другой ответ – 0 баллов

4. Получено верное изображение – 30 баллов
   Каждая неверно проведенная линия – штраф 5 баллов
   4 и более ошибок – 0 баллов

5. Получен верный ответ, приведено разбиение сообщения на кодовые слова - 20 баллов
   Решение в целом верно, но допущена 1 ошибка в ответе – 15 баллов
   Ответ без обоснования - 1 балл
   Ответ без обоснования, но допущена одна ошибка в ответе - 1 баллов
   Решение не удовлетворяет ни одному критерию выше, однако в рассуждениях имеются продвижения – 3 балла
   Любой другой ответ – 0 баллов

Информатика. 3 класс
Решения

1 вариант

1. (7 баллов) Три злых колдуна, каждый с двумя мешками злобных магических штук, хотят переправиться через пропасть на летающем пегасе. На пегаса можно сесть втроем, можно сесть двум колдунам с одним мешком, можно одному колдуну с двумя мешками. Никто из колдунов не доверит свой мешок другим в свое отсутствие, но готов оставить мешок на безлюдном краю пропасти. Смогут ли они переправиться? (Пегас, стоящий на краю пропасти считать частью края пропасти) Если да, то опишите алгоритм их действий. Если нет, покажите, почему это невозможно.
   Решение:
   Да. Обозначим колдунов A, B и C. Сначала C перевозит свои мешки, затем возвращается и перевозит A и B без мешков. После A и C едут за мешками колдуна A, затем B возвращается за своими мешками.

2. (7 баллов) Есть некоторое количество одинаковых квадратных столов. Их нужно расставить для банкета либо буквой П, либо буквой Т. Толщина одной буквы – один стол. В каком случае можно будет посадить больше гостей?
   Решение:
   Если от буквы Т “оторвать” верхнюю планку, появится два посадочных места. Если затем эту планку поставить в одну линию к оставшейся “ножке”, эти два места исчезнут. Аналогично и буква П. Таком образом, нет разницы.
3. (30 баллов) Бутылка масла для лампы выгорает за один час и стоит 60 серебряных монет. пузырёк масла выгорает за 11 минут и стоит 11 серебряных монет. Можно ли отмерить ровно минуту, затратив не более 190 серебряных монет? Можно купить много бутылочек и пузырьков. Масляных ламп неограниченное количество.
   Решение:
   Да. Договоримся лампы, в которые налито масло из бутылки, называть большими, а лампы, в которых налито масло из пузырька, маленькой. Зажжем две большие лампы и одну маленькую. Когда маленькая догорит, зажжем следующую маленькую, и так еще три раза. Когда догорит пятая маленькая лампа, гасим одну большую. Когда догорит вторая большая лампа, одна минута – это промежуток между моментом, когда догорит потушенная ранее большая лампа, и когда сгорит шестая маленькая лампа. Затратим 2*60+6*11=186 серебряных монет..

4. (16 баллов) У Бельчонка есть шесть разных цветных мелков и очень большая площадка, выложенная квадратными плитами. Какое наименьшее число плит

надо покрасить так, чтобы для любых разных цветов нашлась пара плит разных цветов с общей стороной? (Всю площадку красить не обязательно)

Решение:

12. Каждая плита граничит не более чем с четырьмя другими, поэтому каждая плита создает не более четырех пар плит. Для каждого цвета должно быть не менее 5 пар с участием данного цвета. Значит, плит каждого цвета хотя бы по две, всего не меньше 12. Пример:

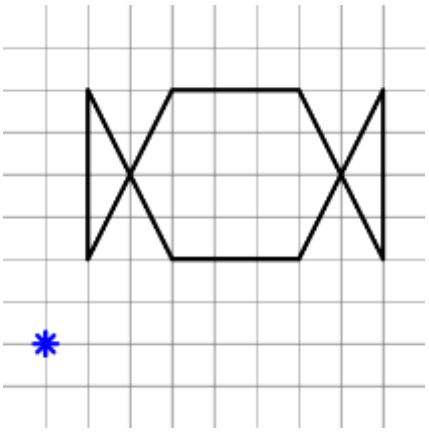
1	2	3
4	5	6
2	3	1
6	4	5

5. (25 баллов) Робот-чертежник умеет, двигаясь по клеточкам, оставлять за собой след в виде линии с помощью пера. У робота имеется следующий набор команд:
   - **опустить перо;**
   - **поднять перо;**
   - **сместиться на(x, y).**

   Команда **сместиться на(x, y)** перемещает робота на **x** вправо или на **x** влево, если число отрицательное, и на **y** вверх или вниз.

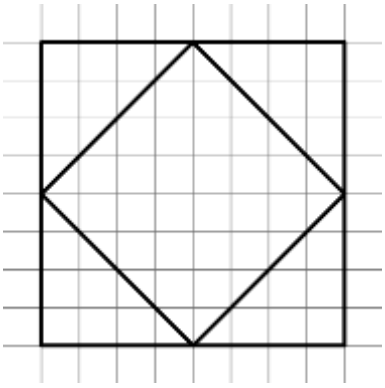
   Например, если в начале работы программы робот стоит в точке обозначенной звездочкой, то в результате выполнения программы:

   сместиться на(1, 2)
   опустить перо
   сместиться на(0, 4)
   сместиться на(2, -4)
   сместиться на(3, 0)
   сместиться на(2, 4)
   сместиться на(0, 4)
   сместиться на(-2, 4)
   сместиться на(-3, 0)
   сместиться на(-2, -4)
   поднять перо



   Получится такой рисунок (справа):

   Составьте программу рисования фигуры, изображенной на рисунке, таким образом, чтобы во время рисования перо не отрывалось от бумаги, и ни одна линия не проводилась дважды. Начальную точку выбрать самостоятельно.



Решение:

опустить перо
сместиться на(0, 4)
сместиться на(8, 0)
сместиться на(0, -8)
сместиться на(-8, 0)
сместиться на(0, 4)
сместиться на(4, 4)
сместиться на(4, -4)
сместиться на(-4, -4)
сместиться на(-4, 4)
поднять перо

6. (15 баллов) Археологи обнаружили зашифрованное сообщение для древнего устройства: **23475554**, а неподалеку клавиатуру от этого устройства:

1	2 абвг	3 дежз
4 ийкл	5 мноп	6 рсту
7 фчцч	8 шщъы	9 ьэюя

Какое слово записано в сообщении?

Решение: белчонок

Информатика. 3 класс
Решения

2 вариант

1. (7 баллов) Три жадных гнома, каждый с двумя мешками золота, хотят переправиться через тоннель на вагонетке. В вагонетку можно сесть втроем, можно сесть двум гномам с одним мешком, можно одному гному с двумя мешками. Никто из гномов не доверит свой мешок другим в свое отсутствие, но готов оставить мешок у безлюдного входа в тоннель. Смогут ли они переправиться? (Вагонетку, стоящую у входа в тоннель с обеих сторон считать частью входа в тоннель) Если да, то опишите алгоритм их действий. Если нет, покажите, почему это невозможно.
   Решение:
   Да. Обозначим гномов A, B и C. Сначала C перевозит свои мешки, затем возвращается и перевозит A и B без мешков. После A и C едут за мешками гнома A, затем B возвращается за своими мешками.

2. (7 баллов) Есть некоторое количество одинаковых квадратных столов. Их нужно расставить для банкета либо буквой П, либо буквой Н. Толщина одной буквы – один стол. В каком случае можно будет посадить больше гостей?
   Решение:
   Если от буквы П “оторвать” левую ножку, появится два посадочных места. Если затем эту планку поставить в одну линию к оставшейся “ножке”, эти два места исчезнут. Аналогично и буква П. Таким образом, нет разницы.

3. (30 баллов) Бутылка масла для лампы выгорает за один час и стоит 120 серебряных монет. пузырёк масла выгорает за 11 минут и стоит 22 серебряные монеты. Можно ли отмерить ровно минуту, затратив не более 190 серебряных монет? Можно купить много бутылочек и пузырьков. Масляных ламп неограниченное количество.
   Решение:
   Да. Договоримся лампы, в которые налито масло из бутылки, называть большими, а лампы, в которых налито масло из пузырька, маленькой. Зажжем две большие лампы и одну маленькую. Когда маленькая догорит, зажжем следующую маленькую, и так еще три раза. Когда догорит пятая маленькая лампа, гасим одну большую. Когда догорит вторая большая лампа, одна минута – это промежуток между моментом, когда догорит потушенная ранее большая лампа, и когда сгорит шестая маленькая лампа. Затратим 2*120+6*22=372 серебряных монеты..
4. (16 баллов) У Бельчонка есть шесть разных красок и очень большая стена, выложенная квадратными плитками. Какое наименьшее число плиток надо

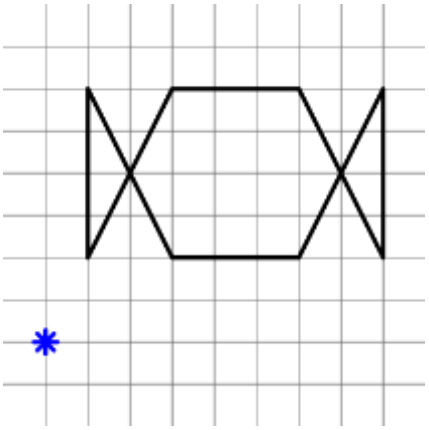
покрасить так, чтобы для любых разных цветов нашлась пара плиток разных цветов с общей стороной? (Всю стену красить не обязательно)
Решение:
12. Каждая плита граничит не более чем с четырьмя другими, поэтому каждая плита создает не более четырех пар плит. Для каждого цвета должно быть не менее 5 пар с участием данного цвета. Значит, плит каждого цвета хотя бы по две, всего не меньше 12. Пример:

1	2	3
4	5	6
2	3	1
6	4	5

5. (25 баллов) Робот-чертежник умеет, двигаясь по клеточкам, оставлять за собой след в виде линии с помощью пера. У робота имеется следующий набор команд:
   - **опустить перо;**
   - **поднять перо;**
   - **сместиться на(x, y).**

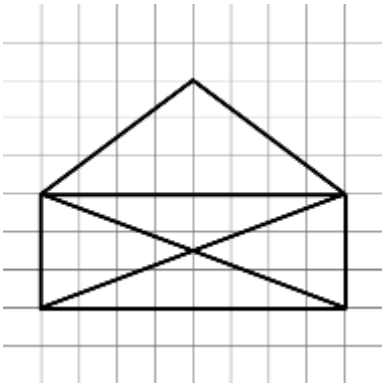
   Команда **сместиться на(x, y)** перемещает робота на **x** вправо или на **x** влево, если число отрицательное, и на **y** вверх или вниз.
   Например, если в начале работы программы робот стоит в точке обозначенной звездочкой, то в результате выполнения программы:

   сместиться на(1, 2)
   опустить перо
   сместиться на(0, 4)
   сместиться на(2, -4)
   сместиться на(3, 0)
   сместиться на(2, 4)
   сместиться на(0, 4)
   сместиться на(-2, 4)
   сместиться на(-3, 0)
   сместиться на(-2, -4)
   поднять перо

   Получится такой рисунок (справа):

   Составьте программу рисования фигуры, изображенной на рисунке, таким образом, чтобы во время рисования перо не отрывалось от бумаги, и ни одна линия не проводилась дважды. Начальную точку выбрать самостоятельно.



Решение:

опустить перо
сместиться на(0, 3)
сместиться на(4, 3)
сместиться на(4, -3)
сместиться на(0, -3)
сместиться на(-8, 3)
сместиться на(8, 0)
сместиться на(-8, -3)
сместиться на(-8, 0)
поднять перо

6. (15 баллов) Археологи обнаружили зашифрованное сообщение для древнего устройства: **544554232**, а неподалеку клавиатуру от этого устройства:

1	2 абвг	3 дежз
4 ийкл	5 мноп	6 рсту
7 фчцч	8 шщъы	9 ьэюя

Какое слово записано в сообщении?

Решение: олимпиада

Информатика. 3 класс
Решения

3 вариант

1. (7 баллов) Три необычайно умных тролля, каждый с двумя любимыми камнями, хотят переправиться через реку на лодке. В лодку можно сесть втроем, можно сесть двум троллям с одним камнем, можно одному одному троллю с двумя камнями. Никто из троллей не доверит свой камень другим в свое отсутствие, но готов оставить камень на безлюдном берегу. Смогут ли они переправиться? (Лодку, стоящую у берегу считать частью берега) Если да, то опишите алгоритм их действий. Если нет, покажите, почему это невозможно.
   Решение:
   Да. Обозначим троллей A, B и C. Сначала C перевозит свои камни, затем возвращается и перевозит A и B без камней. После A и C едут за камнями тролля A, затем B возвращается за своими камнями.

2. (7 баллов) Есть некоторое количество одинаковых квадратных столов. Их нужно расставить для банкета либо буквой Т, либо буквой Н. Толщина одной буквы – один стол. В каком случае можно будет посадить больше гостей?
   Решение:
   Если от буквы Т “оторвать” верхнюю планку, появится два посадочных места. Если затем эту планку поставить в одну линию к оставшейся “ножке”, эти два места исчезнут. Аналогично и буква Н. Таким образом, нет разницы.

3. (30 баллов) Бутылка масла для лампы выгорает за один час и стоит 180 серебряных монет. пузырёк масла выгорает за 11 минут и стоит 33 серебряные монеты. Можно ли отмерить ровно минуту, затратив не более 600 серебряных монет? Можно купить много бутылочек и пузырьков. Масляных ламп неограниченное количество.
   Решение:
   Да. Договоримся лампы, в которые налито масло из бутылки, называть большими, а лампы, в которых налито масло из пузырька, маленькой. Зажжем две большие лампы и одну маленькую. Когда маленькая догорит, зажжем следующую маленькую, и так еще три раза. Когда догорит пятая маленькая лампа, гасим одну большую. Когда догорит вторая большая лампа, одна минута – это промежуток между моментом, когда догорит потушенная ранее большая лампа, и когда сгорит шестая маленькая лампа. Затратим 2*180+6*33=558 серебряных монет.
4. (16 баллов) У Бельчонка есть шесть разных фломастеров и очень большой ватман, расчерченный на квадратики. Какое наименьшее число квадратиков

надо покрасить так, чтобы для любых разных цветов нашлась пара квадратиков разных цветов с общей стороной? (Всю бумагу красить не обязательно)

Решение:

12. Каждый квадратик граничит не более чем с четырьмя другими, поэтому каждый квадратик создает не более четырех пар квадратиков. Для каждого цвета должно быть не менее 5 пар с участием данного цвета. Значит, квадратиков каждого цвета хотя бы по два, всего не меньше 12. Пример:

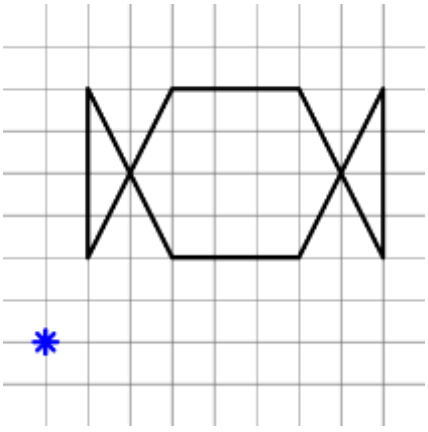
1	2	3
4	5	6
2	3	1
6	4	5

5. (25 баллов) Робот-чертежник умеет, двигаясь по клеточкам, оставлять за собой след в виде линии с помощью пера. У робота имеется следующий набор команд:
   - **опустить перо;**
   - **поднять перо;**
   - **сместиться на(x, y).**

   Команда **сместиться на(x, y)** перемещает робота на **x** вправо или на **x** влево, если число отрицательное, и на **y** вверх или вниз.

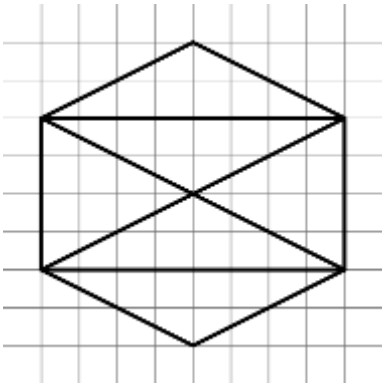
   Например, если в начале работы программы робот стоит в точке обозначенной звездочкой, то в результате выполнения программы:

   сместиться на(1, 2)
   опустить перо
   сместиться на(0, 4)
   сместиться на(2, -4)
   сместиться на(3, 0)
   сместиться на(2, 4)
   сместиться на(0, 4)
   сместиться на(-2, 4)
   сместиться на(-3, 0)
   сместиться на(-2, -4)
   поднять перо

   Получится такой рисунок (справа):

   Составьте программу рисования фигуры, изображенной на рисунке, таким образом, чтобы во время рисования перо не отрывалось от бумаги, и ни одна линия не проводилась дважды. Начальную точку выбрать самостоятельно.



Решение:

опустить перо
сместиться на(0, 4)
сместиться на(4, 2)
сместиться на(4, -2)П546Е4К
сместиться на(-8, -4)
сместиться на(8, 0)
сместиться на(-8, 4)
сместиться на(8, 0)
сместиться на(0, -4)
сместиться на(-4, -2)
сместиться на(-4, 2)
поднять перо

6. (15 баллов) Археологи обнаружили зашифрованное сообщение для древнего устройства: **45756526448**, а неподалеку клавиатуру от этого устройства:

1	2 абвг	3 дежз
4 ийкл	5 мноп	6 рсту
7 фчцч	8 шщъы	9 ьэюя

Какое слово записано в сообщении?

Решение: информатикъ

Информатика. 3 класс
Критерии оценивания

1. Обоснованно получен верный ответ – 7 баллов
   Решение в целом верно, почти удовлетворяет критерию выше, однако имеются неточности – 5 баллов
   Решение не удовлетворяет ни одному критерию выше, однако в рассуждениях имеются продвижения – 2 балла
   Любой другой ответ – 0 баллов
2. Обоснованно получен верный ответ – 7 баллов
   Решение в целом верно, почти удовлетворяет критерию выше, однако имеются неточности – 5 баллов
   Решение не удовлетворяет ни одному критерию выше, однако в рассуждениях имеются продвижения – 2 балла
   Любой другой ответ – 0 баллов
3. Обоснованно получен верный ответ – 30 баллов
   Решение в целом верно, почти удовлетворяет критерию выше, однако имеются неточности – 20 баллов
   Решение не удовлетворяет ни одному критерию выше, однако в рассуждениях имеются продвижения – 5 баллов
   Дан верный ответ без обоснования – 3 балла
   Любой другой ответ – 0 баллов
4. Обоснованно получен верный ответ – 16 баллов
   Решение в целом верно, почти удовлетворяет критерию выше, однако имеются неточности – 10 баллов
   Решение не удовлетворяет ни одному критерию выше, однако в рассуждениях имеются продвижения – 5 баллов
   Дан верный ответ без обоснования – 3 балла
   Любой другой ответ – 0 баллов
5. Программа написана верно, отсутствуют накладывающиеся линии - 25 баллов
   Программа написана в целом верно, но присутствуют лишние операции или одна линия накладывается - 20 баллов
   Программа написана в целом верно, но накладываются 2 линии - 10 баллов
   Есть продвижения по решению - 3 баллов
   Любое другое решение - 0 баллов
6. Получено обоснованное решение - 15 баллов
   Допущена несущественная ошибка в логике рассуждения - 10 баллов
   Ответ без обоснования - 5 баллов
   Есть продвижения по решению - 3 балла
   Любое другое решение - 0 баллов

Информатика. 4 класс
Решения

1 вариант

1. (7 баллов) На дощечке написаны 2 числа: слева 2023, справа 1000. За один ход можно прибавить к числу, написанному слева натуральное число, а число справа умножить на то же самое число. Можно ли уравнять числа на дощечке, сделав не более 1000 ходов?
   Решение:
   Можно. Прибавим к 2023 число 3, а 1000 умножим на 3. Получим 2026 и 3000. Затем за 974 ходов мы сможем уравнять числа.
2. (25 баллов) Бикфордов шнур горит равномерно. Длинный бикфордов шнур горит одну минуту и стоит 60 тугриков. Короткий бикфордов шнур сгорает за 11 секунд и стоит 11 тугриков. Можно ли отмерить ровно одну секунду, затратив не более 150 тугриков? Если да, запишите алгоритм действий.
   Решение:
   Да, зажжем один длинный и один короткий шнур. Когда короткий догорит, зажжем следующий короткий, и так еще три раза. Когда сгорит пятый короткий шнур, зажжем сразу два коротких и погасим один из их в тот момент, когда догорит большой. Тем самым получим два огарка, каждый из которых рассчитан на 6 секунд. Зажжем один короткий шнур и последовательно эти два огарка. Секунда – промежуток времени между моментом, когда догорит восьмой короткий шнур, и моментом, когда догорит второй огарок шнура. Потратим 60+8*11=148 тугриков.
3. (10 баллов) Бельчонок делает запасы на зиму, раскладывая 100 орехов на 10 сундучков, разное число орехов в каждый. Может ли он разложить орехи так, что орехи ни из одного сундучка нельзя частично переложить в одиннадцатый так, чтобы число орехов во всех сундучках оставалось различным?
   Решение:
   Разложим орехи в 1, 3, 5, …, 19 орехов. Если одну из кучек поделить пополам, нечетная часть кучки даст повторение.
4. (20 баллов) Девять бельчат играли на поляне: они становились в клетки квадрата 3х3, разбегались, затем вставали обратно, и так три раза. Каждый раз бельчата, оказавшиеся в соседних по стороне клетках, обменивались открыточками. Правда ли, что какие-то два бельчонка так и не обменялись открытками?
   Решение:
   Раскрасим квадрат в шахматном порядке. Каждый раз обмениваются открытками бельчата, стоящие на клетках разного цвета. В первый раз на каком-то цвете стояли не менее 5 бельчат, они не обменялись открытками. Из них не менее 3х во второй раз стояли на одном цвете и не обменялись.

Из этих бельчат какие-то два стояли на одном цвете и не обменялись открытками. Иначе: разобьем бельчат на такие группы: ччч, ччб,…, ббб (например, в группу чбч входят гномы, которые в первый раз стояли на черном, во второй – на белом, в третий – на черном). Таких групп 8, бельчат 9, групп меньше, значит, в какой-то группе не меньше 2х бельчат. Они не обменялись открытками.

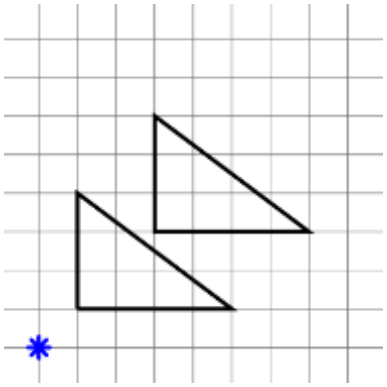
5. (20 баллов) Робот-чертежник умеет, двигаясь по клеточкам, оставлять за собой след в виде линии с помощью пера. У робота имеется следующий набор команд:
   - **опустить перо;**
   - **поднять перо;**
   - **сместиться на(x, y);**
   - **повтори к нц**
     …
     кц

Команда **сместиться на(x, y)** перемещает робота на **x** вправо или на **x** влево, если число отрицательное, и на **y** вверх или вниз.

Команда **повтори к нц … кц** нужна для рисования повторяющихся элементов, с помощью нее робот повторяет **к** раз все действия стоящие вместо **….**

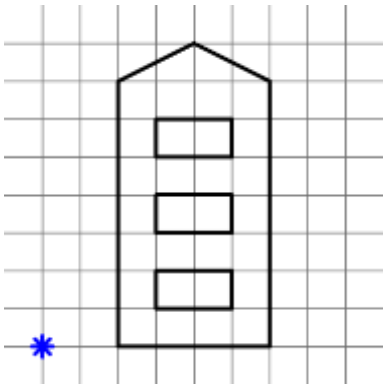
Например, если в начале работы программы робот стоит в точке обозначенной звездочкой, то в результате выполнения программы:

1. **сместиться на(1, 1)**
2. **повтори 2 нц**
   опустить перо
   сместиться на(0, 3)
   сместиться на(4, -3)
   сместиться на(-4, 0)
   поднять перо
   сместиться на(2, 2)
   кц
3. **поднять перо**



Получится такой рисунок (справа):

Напишите максимально короткую программу для робота, с использованием команды **повтори** и без накладывающихся линий, чтобы в результате получился такой рисунок (справа):



Решение:

1. **сместиться на(2, 0)**
2. **опустить перо**
3. **сместиться на(0, 7)**
4. **сместиться на(2, 1)**
5. **сместиться на(2, -1)**

6. сместиться на(0, -7)
7. сместиться на(-4, 0)
8. поднять перо
9. сместиться на(1, 1)
10. повтори 3 нц
опустить перо
сместиться на(0, 1)
сместиться на(2, 0)
сместиться на(0, -1)
сместиться на(-2, 0)
поднять перо
сместиться на(0, 2)
кц

6. (18 баллов) Для кодирования некоторого сообщения, состоящего только из букв Б, Е, Л, К, решили использовать неравномерный код такой, что никакой код буквы не является началом кода другой буквы. Для буквы Б использовали код 0, для буквы Л - 10. Какова наименьшая возможная суммарная длина всех четырех кодов?

Решение:

Найдем наиболее короткие представления для всех букв. Кодовые слова 01 и 00 использовать нельзя, поскольку тогда нарушается условие. Используем, например, для буквы Е кодовое слово 11. Тогда для четвёртой буквы нельзя подобрать кодовое слово, не нарушая условие. Следовательно, для оставшихся двух букв нужно использовать трёхзначные кодовые слова. Закодируем буквы Е и К кодовыми словами 110 и 111. Тогда суммарная длина всех четырёх кодовых слов равна 1 + 2 + 3 + 3 = 9.

Информатика. 4 класс
Решения

2 вариант

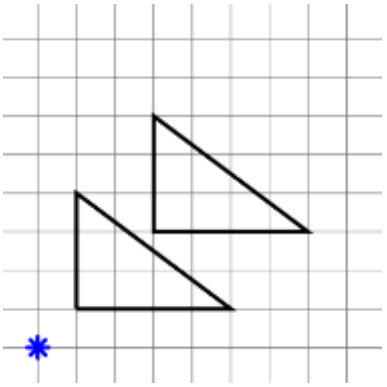
1. (7 баллов) На школьной доске написаны 2 числа: слева 2023, с правой 1001. За один ход можно прибавить к числу, написанному слева натуральное число, а число справа умножить на то же самое число. Можно ли уравнять числа на дощечке, сделав не более 1000 ходов?
   Решение:
   Можно. Прибавим к 2023 число 3, а 1001 умножим на 3. Получим 2026 и 3003. Затем за 977 ходов мы сможем уравнять числа.
2. (25 баллов) Бикфордов шнур горит равномерно. Длинный бикфордов шнур горит одну минуту и стоит 120 тугриков. Короткий бикфордов шнур сгорает за 11 секунд и стоит 22 тугрика. Можно ли отмерить ровно одну секунду, затратив не более 300 тугриков? Если да, запишите алгоритм действий.
   Решение:
3. Да, зажжем один длинный и один короткий шнур. Когда короткий догорит, зажжем следующий короткий, и так еще три раза. Когда сгорит пятый короткий шнур, зажжем сразу два коротких и погасим один из их в тот момент, когда догорит большой. Тем самым получим два огарка, каждый из которых рассчитан на 6 секунд. Зажжем один короткий шнур и последовательно эти два огарка. Секунда – промежуток времени между моментом, когда догорит восьмой короткий шнур, и моментом, когда догорит второй огарок шнура. Потратим 120+8*22=296 тугриков.
4. (10 баллов) Бельчонок делает запасы на зиму, раскладывая 100 орехов на 10 кучек, разное число орехов в каждой. Может ли он разложить орехи так, что ни одну из кучек нельзя разбить на две так, чтобы число орехов во всех полученных кучках оставалось различным?
   Решение:
   Разложим орехи в 1, 3, 5, …, 19 орехов. Если одну из кучек поделить пополам, нечетная часть кучки даст повторение
5. (20 баллов) Девять бельчат играли на поляне: они становились в клетки квадрата 3х3, разбегались, затем вставали обратно, и так три раза. Каждый раз бельчата, оказавшиеся в соседних по стороне клетках, обнимались. Правда ли, что какие-то два бельчонка так и не обнялись?
   Решение:
   Раскрасим квадрат в шахматном порядке. Каждый раз обнимаются бельчата, стоящие на клетках разного цвета. В первый раз на каком-то цвете стояли не менее 5 бельчат, они не обнимались. Из них не менее 3х во второй раз стояли на одном цвете и не обнимались. Из этих бельчат какие-то два стояли на одном цвете и не обнимались. Иначе: разобьем бельчат на такие группы: ччч, ччб,…, ббб (например, в группу чбч входят гномы, которые в

первый раз стояли на черном, во второй – на белом, в третий – на черном). Таких групп 8, бельчат 9, групп меньше, значит, в какой-то группе не меньше 2х бельчат. Они не обнимались.

5. (20 баллов) Робот-чертежник умеет, двигаясь по клеточкам, оставлять за собой след в виде линии с помощью пера. У робота имеется следующий набор команд:
   - **опустить перо;**
   - **поднять перо;**
   - **сместиться на(x, y);**
   - **повтори к нц**
     …
     кц

   Команда **сместиться на(x, y)** перемещает робота на **x** вправо или на **x** влево, если число отрицательное, и на **y** вверх или вниз.
   Команда **повтори к нц … кц** нужна для рисования повторяющихся элементов, с помощью нее робот повторяет **к** раз все действия стоящие вместо **….**
   Например, если в начале работы программы робот стоит в точке обозначенной звездочкой, то в результате выполнения программы:

   1. **сместиться на(1, 1)**
   2. **повтори 2 нц**
      опустить перо
      сместиться на(0, 3)
      сместиться на(4, -3)
      сместиться на(-4, 0)
      поднять перо
      сместиться на(2, 2)
      кц
   3. **поднять перо**



   Получится такой рисунок (справа):
   Напишите максимально короткую программу для робота, с использованием команды **повтори** и без накладывающихся линий, чтобы в результате получился такой рисунок:

Решение:

11.сместиться на(3.5, 1)
12.опустить перо
13.сместиться на(0, -1)

14.сместиться на(1, 0)
15.сместиться на(0, 1)
16.поднять перо
17.сместиться на(1.5, 0)
18.повтори 3 нц
опустить перо
сместиться на(-2, 2)
сместиться на(-2, -2)
сместиться на(4, 0)
поднять перо
сместиться на(0, 2)
кц

6. (18 баллов) Для кодирования некоторого сообщения, состоящего только из букв А, Б, В, Г, решили использовать неравномерный код такой, что никакой код буквы не является началом кода другой буквы. Для буквы Б использовали код 1, для буквы В - 011. Какова наименьшая возможная суммарная длина всех четырех кодов?

Решение:

Найдем наиболее короткие представления для всех букв. Кодовые слова 1 и 01, и все слова начинающиеся с 1 использовать нельзя, поскольку тогда нарушается условие. Используем, например, для буквы А кодовое слово 00. Тогда для четвёртой буквы самое короткое кодовое слово будет 010. Тогда суммарная длина всех четырёх кодовых слов равна 1 + 2 + 3 + 3 = 9.

Информатика. 4 класс
Решения

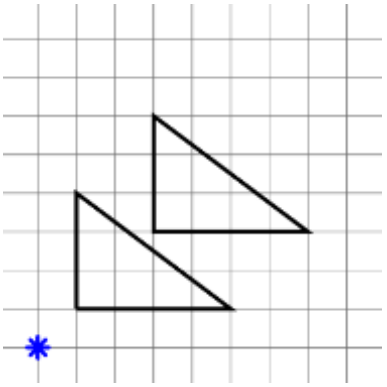
3 вариант

1. На асфальте написаны 2 числа: слева 2023, с правой 1002. За один ход можно прибавить к числу, написанному слева натуральное число, а число справа умножить на то же самое число. Можно ли уравнять числа на асфальте, сделав не более 1000 ходов?
   Решение:
   Можно. Прибавим к 2023 число 3, а 1002 умножим на 3. Получим 2026 и 3006. Затем за 980 ходов мы сможем уравнять числа.
2. Бикфордов шнур горит равномерно. Длинный бикфордов шнур горит одну минуту и стоит 120 тугриков. Короткий бикфордов шнур сгорает за 11 секунд и стоит 22 тугрика. Можно ли отмерить ровно одну секунду, затратив не более 300 тугриков? Если да, запишите алгоритм действий.
   Решение:
   Да, зажжем один длинный и один короткий шнур. Когда короткий догорит, зажжем следующий короткий, и так еще три раза. Когда сгорит пятый короткий шнур, зажжем сразу два коротких и погасим один из их в тот момент, когда догорит большой. Тем самым получим два огарка, каждый из которых рассчитан на 6 секунд. Зажжем один короткий шнур и последовательно эти два огарка. Секунда – промежуток времени между моментом, когда догорит восьмой короткий шнур, и моментом, когда догорит второй огарок шнура. Потратим 120+8*22=296 тугриков.
3. Бельчонок делает запасы на зиму, раскладывая 100 орехов на 10 мешочков, разное число орехов в каждый. Может ли он разложить орехи так, что орехи ни из одного мешочка нельзя частично переложить в одиннадцатый так, чтобы число орехов во всех мешочках оставалось различным?
   Решение:
   Разложим орехи в 1, 3, 5, …, 19 орехов. Если одну из кучек поделить пополам, нечетная часть кучки даст повторение
4. Девять бельчат играли на поляне: они становились в клетки квадрата 3х3, разбегались, затем вставали обратно, и так три раза. Каждый раз бельчата, оказавшиеся в соседних по стороне клетках, пожимали друг другу лапы. Правда ли, что какие-то два бельчонка так и не пожали друг другу лапы?
   Решение:
   Раскрасим квадрат в шахматном порядке. Каждый раз здороваются бельчата, стоящие на клетках разного цвета. В первый раз на каком-то цвете стояли не менее 5 бельчат, они не здоровались. Из них не менее 3х во второй раз стояли на одном цвете и не здоровались. Из этих бельчат какие-то два стояли на одном цвете и не здоровались. Иначе: разобьем бельчат на такие группы: ччч, ччб,…, ббб (например, в группу чбч входят гномы, которые в

первый раз стояли на черном, во второй – на белом, в третий – на черном). Таких групп 8, бельчат 9, групп меньше, значит, в какой-то группе не меньше 2х бельчат. Они не здоровались

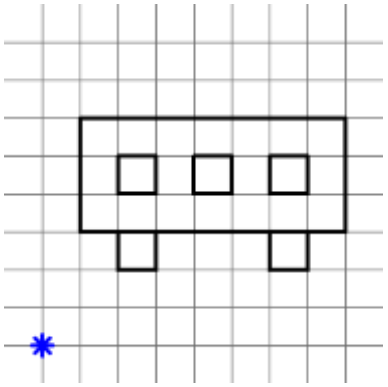
5. Робот-чертежник умеет, двигаясь по клеточкам, оставлять за собой след в виде линии с помощью пера. У робота имеется следующий набор команд:
   - **опустить перо;**
   - **поднять перо;**
   - **сместиться на(x, y);**
   - **повтори к нц**
     …
     кц

   Команда **сместиться на(x, y)** перемещает робота на **x** вправо или на **x** влево, если число отрицательное, и на **y** вверх или вниз.
   Команда **повтори к нц … кц** нужна для рисования повторяющихся элементов, с помощью нее робот повторяет **к** раз все действия стоящие вместо **….**
   Например, если в начале работы программы робот стоит в точке обозначенной звездочкой, то в результате выполнения программы:



   1. **сместиться на(1, 1)**
   2. **повтори 2 нц**
      опустить перо
      сместиться на(0, 3)
      сместиться на(4, -3)
      сместиться на(-4, 0)
      поднять перо
      сместиться на(2, 2)
      кц
   3. **поднять перо**

   Получится такой рисунок (справа):



   Напишите максимально короткую программу для робота, с использованием команды **повтори** и без накладывающихся линий, чтобы в результате получился такой рисунок:

   Решение:
   1. **сместиться на(1, 3)**
   2. **опустить перо**
   3. **сместиться на(0, 3)**
   4. **сместиться на(7, 0)**
   5. **сместиться на(0, -3)**
   6. **сместиться на(-7, 0)**
   7. **поднять перо**
   8. **сместиться на(1, 0)**
   9. **опустить перо**

10.сместиться на(0, -1)
11.сместиться на(1, 0)
12.сместиться на(0, 1)
13.поднять перо
14.сместиться на(3, 0)
15.опустить перо
16.сместиться на(0, -1)
17.сместиться на(1, 0)
18.сместиться на(0, 1)
19.поднять перо
20.сместиться на(0, 1)
21.повтори 3 нц
опустить перо
сместиться на(0, 1)
сместиться на(-1, 0)
сместиться на(0, -1)
сместиться на(1, 0)
поднять перо
сместиться на(2, 0)
кц

6. Для кодирования некоторого сообщения, состоящего только из букв Ж, З, И, К, решили использовать неравномерный код такой, что никакой код буквы не является началом кода другой буквы. Для буквы Ж использовали код 0, для буквы З - 110. Какова наименьшая возможная суммарная длина всех четырех кодов?

Решение:

Найдем наиболее короткие представления для всех букв. Кодовые слова 0 и 11, и все слова начинающиеся с 0 использовать нельзя, поскольку тогда нарушается условие. Используем, например, для буквы И кодовое слово 10. Тогда для четвёртой буквы самое короткое кодовое слово будет 111. Тогда суммарная длина всех четырёх кодовых слов равна 1 + 2 + 3 + 3 = 9.

Информатика. 4 класс
Критерии оценивания

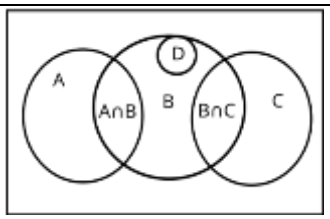
1. Обоснованно получен верный ответ – 7 баллов
   Приведено в целом верное решение, но имеются неточности – 5 баллов
   Дан верный ответ “да” без обоснования – 1 балл
   Решение не удовлетворяет ни одному критерию выше, однако в рассуждениях имеются продвижения – 1 балл
   Любой другой ответ – 0 баллов
2. Обоснованно получен верный ответ – 25 баллов
   Приведено в целом верное решение, но имеются неточности – 20 баллов
   Решение не удовлетворяет ни одному критерию выше, однако в рассуждениях имеются продвижения – 10 баллов
   Дан верный ответ “да” без обоснования – 3 балла
   Любой другой ответ – 0 баллов
3. Обоснованно получен верный ответ – 10 баллов
   Приведено в целом верное решение, но имеются неточности – 7 баллов
   Решение не удовлетворяет ни одному критерию выше, однако в рассуждениях имеются продвижения – 3 балла
   Дан верный ответ “да” без обоснования – 1 балл
   Любой другой ответ – 0 баллов
4. Обоснованно получен верный ответ – 20 баллов
   Приведено в целом верное решение, но имеются неточности – 15 баллов
   Приведено в целом верное решение, но имеется критическая ошибка – 7 баллов
   Решение не удовлетворяет ни одному критерию выше, однако в рассуждениях имеются продвижения – 7 баллов
   Дан верный ответ “да” без обоснования – 1 балл
   Любой другой ответ – 0 баллов

5. Программа написана верно, отсутствуют лишние операции, использована команда повтори - 20 баллов
   Программа написана в целом верно, использована команда повтори, но присутствуют лишние операции или одна линия накладывается - 15 баллов
   Программа написана в целом верно, не используется операция повтори или накладываются 2 линии - 10 баллов
   Есть продвижения по решению - 3 баллов
   Любое другое решение - 0 баллов
6. Получено обоснованное решение, указаны коды всех букв - 18 баллов
   Допущена несущественная ошибка в логике рассуждения - 15 баллов
   Ответ без обоснования - 5 баллов
   Есть продвижения по решению - 3 балла

Любое другое решение - 0 баллов

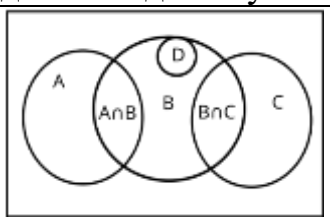
Информатика. 5 класс
Решения

1 вариант

№	Правильный ответ	Балл	Прим.
1.	Найдем количество шестизначных четных чисел палиндромов. Можно рассмотреть только первые три позиции, на которые можно поставить цифры. Первая должна быть четной т.е. может быть 0,2,4,6,8, но так как последним нуль не может быть получаем 4*10*10 вариантов. Рассмотрим кол-во вариантов четных четырехзначных палиндромов в двоичной системе счисления(т.е. если числа состоят только из 0 и 1). Т.к. число четное, то первым должен идти нуль, но по условию число 0110 – не четырехзначное, получаем таких чисел нет. **Ответ: 400**.	20	
2.	(А)**10** (Б)**17** (В)**12,14.** Сравнения при поиске числа 14: -1-е сравнение Р:17 Б:10 -2-е сравнение Р:14 Б:14 Г) Число элементов при поиске каждый раз уменьшается вдвое отсюда максимальное число ходов, которое может сделать Бельчонок равняется 3-м так как каждый раз число возможных вариантов постоянно уменьшается вдвое, отсюда он может совершить три хода если это числа 17,12,8,3. И при этом робот должен совершить тоже максимальное число ходов отсюда получаем число **3**.	23	
3.	 $A = \{\text{Каладиум}\} = 23;$ $B = \{\text{Бегония}\} = 27;$ $C = \{\text{Калатея}\} = 24;$ $D = \{\text{Гинура }\} = 1.$ $A \cap B = 3; B \cap C = 4; D \cap B = 1$ Всего дали ответ $\text{A} + \text{B} + \text{C} - A \cap B - B \cap C = 67.$ Значит ничего не ответили $2090 - 67 = 2023$ человека **Ответ:2023**	20	
4.	**Ответ: верные утверждения: А1, Б1, В2. Числа: 4,4,4** Если предположить, что А2 – верно, то Б1 и Б2 оба неверны, но по условию хотя бы одно утверждение верно значит А1 – верно. Пуст Б2 – верно, тогда цифры А, В, С – 2,4,8 или 2,8,4 соответственно, но тогда утверждения В1 и В2 неверны. Приходим к выводу, что верны утверждения А1, Б1, В2. Далее находим числа: 4,4,4	17	
5.	**Ответ: бельчонок.**	20	

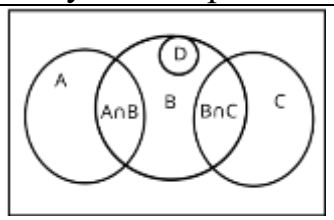
	Рассмотрим все возможные случаи. Начнем со второго шифра предположим, что шифровка велась именно им, тогда ЯЫФДИСТСХ получилось простым сдвигом каждой буквы слова на одну позицию вправо (или в случае с Я влево). Тогда ЯЫФДИСТСХ <= ЮЪУГЗРСРФ. Применим к этому набору букв первый алгоритм шифрования: для того чтобы получить Ю (32) нужно решить уравнение 33-i+1=32 => i=2 (Б). По аналогии решаем остальные и получаем слово БЕЛЬЧОНОК.		

2 вариант

№	Правильный ответ	Балл	Прим.
1.	Найдем количество семизначных четных чисел палиндромов. Можно рассмотреть только первые четыре позиции, на которые можно поставить цифры. Первая должна быть четной т.е. может быть 0,2,4,6,8, но так как последним нуль не может быть получаем 4*10*10*10 вариантов. Рассмотрим кол-во вариантов четных пятизначных палиндромов в двоичной системе счисления(т.е. если числа состоят только из 0 и 1). Т.к. число четное, то первым должен идти нуль, но по условию это не является числом, получаем, что таких чисел нет. **Ответ: 4000.**	20	
2.	(А)**10** (Б)**3** (В)**4,8.** Сравнения при поиске числа 4: -1-е сравнение Р:3 Б:10 -2-е сравнение Р:4 Б:4 Г) Число элементов при поиске каждый раз уменьшается вдвое отсюда максимальное число ходов, которое может сделать Бельчонок равняется 3-м, так как после каждого хода число возможных вариантов уменьшается вдвое, этим ходам соответствуют числа 3,8,25,149. И при этом робот должен совершить тоже максимальное число ходов отсюда получаем число **149**	23	
3.	 $A = \{\text{Теннис}\} = 42;$ $B = \{\text{Тетрис}\} = 51;$ $C = \{\text{Футбол}\} = 27;$ $D = \{\text{Гольф }\} = 2.$ $A \cap B = 5; B \cap C = 7; D \cap B = 2$ Всего дали ответ $\text{A} + \text{B} + \text{C} - A \cap B - B \cap C = 108.$ Значит ничего не ответили $2130 - 108 = 2022$ человека **Ответ:2022**	20	
4.	**Ответ: верные утверждения: А1, Б1, В1. Числа: 3,5,5** Если предположить, что А2 – верно, то верно Б2 это значит, что А = 4 и В=3, С=9 или В=9, С=3, но тогда В1 и	17	

	В2 – неверны, чего не может быть значит А1 и Б1 верны отсюда верно и В1. Значит А=3, В=5, С=5.		
5.	**Ответ: ОЛИМПИАДА.** Рассмотрим все возможные случаи. Начнем со первого шифра предположим, что шифровка велась именно им, тогда первая букву слова СФЧУРЧАЬА можем найти из уравнения 33-i+1=19 значит i=15 (Н) по аналогии поступаем со всеми буквами и получаем слово НКЗЛОЗЯГЯ. Далее посмотрим на второй шифр и поймем как работает он допустим для буквы О(16) он поставит в соответствие букву Н(15). Рассуждаем также далее получим слово ОЛИМПИАДА.	20	

3 вариант

№	Правильный ответ	Балл	Прим.
1.	Найдем количество шестизначных четных чисел палиндромов. Можно рассмотреть только первые три позиции, на которые можно поставить цифры. Первая должна быть четной т.е. может быть 0,2,4,6,8, но так как последним нуль не может быть получаем $4 * 10 * 10$ вариантов. Рассмотрим кол-во вариантов четных пятизначных чисел которые состоят из 0 и 1. Т.к. число четное, то первым должен идти нуль, а последним 1(если смотреть справа налево) получаем следующее (1 _ _ _ 0) на каждое их трех свободных мест можно поставить по две цифры отсюда получаем $1 * 2 * 2 * 2 * 1$ или $2^3 = 8$ **Ответ: 408**.	20	
2.	(А)**100** (Б)**114** (В)**104,108.** Сравнения при поиске числа 104: -1-е сравнение Р:114 Б:100 -2-е сравнение Р:104 Б:104 Г) Число элементов при поиске каждый раз уменьшается вдвое отсюда максимальное число ходов, которое может сделать Робот равняется 3-м, так как после каждого хода число возможных вариантов уменьшается вдвое, значит за максимальное число ходов Робот может совершить три хода если это числа 108,145,199. Но так как Бельчонок должен совершить тоже максимальное число ходов, то будет выбрано число **199**.	23	
3.	 $A = \{\text{Теннис}\} = 80;$ $B = \{\text{Тетрис}\} = 70;$ $C = \{\text{Футбол}\} = 60;$ $D = \{\text{Гольф }\} = 2.$ $A \cap B = 20; B \cap C = 20; D \cap B = 2$	20	

	Всего дали ответ $A + B + C - A \cap B - B \cap C = 170$. Значит ничего не ответили $2450 - 170 = 2280$ человека **Ответ:2280**		
4.	**Ответ: верные утверждения: А2, Б2, В2. Числа: 9,9,9** Если предположить, что А1 – верно, то верно Б1 или Б2 пусть Б1 верно это значит, что А = 9 и В=9, С=7 или В=7, С=9, но тогда В1 и В2 – неверны, чего не может быть аналогично проверяем Б2 и приходим к выводу, что верно А2 и Б2 далее находим, что А= 9 В = 9 С = 9.	17	
5.	**Ответ: ИНФОРМАТИК.** Рассмотрим все возможные случаи. Начнем с первого шифра предположим, что шифровка велась именно им, тогда первая букву слова ТНЖМКОЫИТР можем найти из уравнения 33-i+1=20 значит i=14 (М) по аналогии поступаем со всеми буквами и получаем слово МСШТФРДЦМО. Далее посмотрим на второй шифр и поймем как работает он допустим для буквы И(10) он поставит в соответствие букву М(14). Рассуждаем также далее получим слово ИНФОРМАТИК	20	

Информатика. 5 класс
Критерии оценивания

1 Задание.

1.1. Верно получено число чисел палиндромов в десятичной системе счисления – 12 баллов.

1.2. При рассмотрении палиндромов в десятичной системе счисления учитываются числа с незначащими нулями – снимается 6 баллов.

1.3. Верно посчитано число чисел, которые состоят из нулей и единиц– 8 баллов.

1.4. В ходе решения допущена арифметическая ошибка, повлиявшая на ответ при исправлении которой выполняются условия п. 1.1. и п. 1.3. – ставится 16 баллов.

1.5. Получены верный обоснованный ответ и решение которые соответствуют пунктам 1.1 и 1.3 – 20 баллов.

2 Задание.

2.1. Получен верный ответ в пункте (А) – 2 балла.

2.2. Получен верный ответ в пункте (Б) – 2 балла.

2.3. В пункте (В) выписаны два числа и присутствует пояснение – 6 баллов.

2.4. В пункте (В) выписаны только числа без пояснений – 3 балл.

2.5. В пункте (В) выписано одно число и присутствует пояснение – 4 балла

2.6. В пункте (В) выписано только одно число без пояснения – 2 балла

2.7. В пункте (Г) выписано число, и присутствует пояснение в том числе полный перебор – 9 баллов.

2.8. В пункте (Г) показано, что Робот может совершить не более 3-х ходов – добавляется 4 балла.

2.9 В пункте (Г) выписано только число без полного пояснения – 2 балла

2.10. Получены верный обоснованный ответ и решение которые соответствуют пунктам 2.1, 2.2, 2.3, 2.7, 2.8. – 23 балла

3 Задание

3.1. Дан верный ответ без полного обоснования – 5 баллов.

3.2. В решении указано скольким людям нравится только один вид спорта или цветок, и при этом присутствует обоснование – 12 баллов

3.3. Верно посчитано число людей, давших ответ – 16 баллов

3.4. Допущена арифметическая ошибка, которая повлекла за собой ошибку при этом ход решения верный и в случае исправления ошибки ответ будет верным – 17 баллов.

3.5. Получен верный обоснованный ответ – 20 баллов.

4 Задание

4.1. Верно отобраны все три утверждения, выписаны числа и при этом нет обоснования почему эти числа подходят по условию – 7 баллов.

4.2. Получен верный обоснованный ответ 17 баллов.

5 Задание

5.1. Получен верный ответ без обоснования – 1 балл.

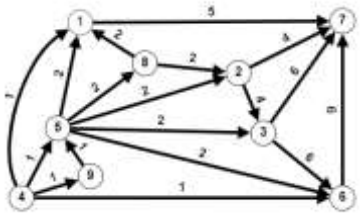
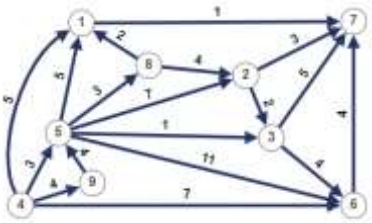
5.2. Если в процессе решения было единожды дешифровано слово с использованием одного из шифров при этом не важно привело это к верному ответу или нет и при этом допущено не больше одной ошибки – 10 баллов.

5.3. Было показано что именно делает каждый из шифров по отдельности (сдвиги) – 15 баллов.
5.4. Получен верный обоснованный ответ – 20 балла.

Информатика. 6 класс
Решения

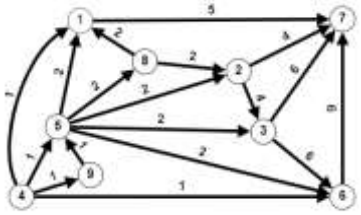
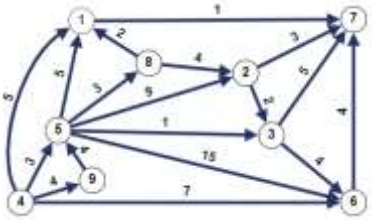
1 вариант

№	Правильный ответ	Балл
1	Например, можно сделать так: $\frac{2*2023+2023*2021}{2023*2023}$. По шагам: 1. Умножим число 2 на 2023 получаем $2 * 2023$ 2. Прибавим к числу 2*2023, 2021 раз число 2023: $2 * 2023 + 2023 * 2021$ 3. Поделим получившиеся число на 2023 дважды. Далее напишем, как можно получить из 1 число 2. Например так $\frac{1*2023+2023}{2023}$ По шагам: 1. Умножим единицу на 2023 2. Прибавим 2023 3. Поделим на 2023	15
2	Так как стоимость одинакова рассмотрим среднюю скорость. В случае отправки файла размером в 54 Кбайт убедимся, что “Омега-23” отправит этот файл за 9 минут (54/6=9). Найдем какая средняя скорость отправки файла в случае, если выбрать “Бета-23” $\frac{(0+1+1+2+3+5+8+13+21)}{9} = 6$. Отсюда можем сделать вывод что можно выбрать любую компанию так как время отправки будет одно и тоже (и цены одинаковы). Отсюда можно сделать вывод что если файл больше, чем 54 Кбайт, то средняя скорость “Бета-23” будет больше, чем средняя скорость “Омега-23”, а значит интернет от этой компании выгоднее в случае, если размер файла больше 54 Кбайт.	20
3	Далее будем считать, что все простые числа, которые написаны на доске больше 2. Рассмотрим числа на первой доске. Если вычесть или просуммировать два простых числа, то в результате будет четное число. Так как $\text{нечет.} \pm \text{нечет.} = \text{чет.}$ получим, что в любом случае на первой доске будет нечетное число так как чисел 2023. Рассмотрим числа на второй доске. На четные числа мы можем не обращать внимания они не влияют на конечное значение, а число нечетных нечетно значит в итоге получим нечетное число. Отсюда на двух досках в итоге будут написаны два нечетных числа, значит всегда в конце игре будет четное число. Отсюда приходим к выводу, что ни ходы Ивана, ни ходы Иннокентия не влияют на результат игры, а поэтому мы можем привести совершенно любую тактику.	20

	Если предположить что на первой доске было написано нечетное число раз число 2, то получим ситуацию чет и нечет для которой нет универсальной тактики.	
4	  Нарисуем для наглядности граф. С помощью него посчитаем число путей. А именно можно посчитать сколько путей ведут до каждой вершины начиная от 4. Всего путей **24**. (см рис. 1) Для поиска кратчайшего пути ищем минимальный путь до “5” – 3. Из этого следует, что расстояние до 7 в любом случае будет не меньше 4. Так как до “8” кратчайший путь составляет 6, то значит меньше 6 расстояния не может быть. Значит самый короткий путь **417(длина 6)**.	25
5	1) Программа выведет “да” 2) Программа выведет “нет” 3) Этот алгоритм проверяет число на простоту. Выводит “нет”, если число составное, а “да” если простое. Каждую итерацию k сравнивается с number если k меньше number, то мы проверяем делится ли number на k. Причем заметим, что number находится в промежутке в промежутке от 2 до number-1 т.е. Это означает что тем самым мы ищем делитель числа number отличный от единицы и самого числа number. Причем после того, как k становится равным number, и у него не найдено делителя, то выводится “да”, т.е. если число number простое. Из этих рассуждений делаем вид, что данный алгоритм предназначен для проверки числа на простоту. Т.е. элементарный тест на простоту.	20

2 вариант

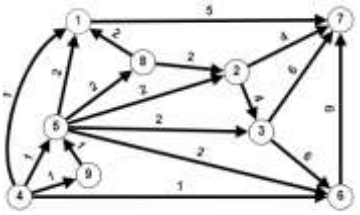
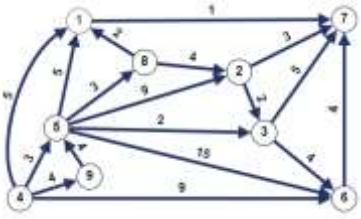
№	Правильный ответ	Балл
1	Например, можно сделать так: $\frac{2*2024+2024*2022}{2024*2024}$. По шагам: 1. Умножим число 2 на 2024 получаем $2 * 2024$ 2. Прибавим к числу 2*2024, 2022 раза число 2024: $2 * 2024 + 2024 * 2022$ 3. Поделим получившиеся число на 2024 дважды. Далее напишем, как можно получить из 1 число 2. Например так $\frac{1*2024+2024}{2024}$ По шагам: 1. Умножим единицу на 2024 2. Прибавим 2024	15

	3. Поделим на 2024	
2	Так как стоимость одинакова рассмотрим среднюю скорость. В случае отправки файла размером в 15 Кбайт убедимся, что “Дзета-23” отправит этот файл за 5 минут (15/3=5). Найдем какая средняя скорость отправки файла в случае, если выбрать “Альфа-23” $\frac{(1+1+2+4+7)}{5} = 3$. Отсюда можем сделать вывод что можно выбрать любую компанию так как время отправки будет одно и тоже. Отсюда можно сделать вывод что если файл больше, чем 15 Кбайт, то средняя скорость “Альфа-23” будет больше, чем средняя скорость “Омега-23”, а значит интернет от этой компании выгоднее в случае, если размер файла больше 15 Кбайт.	20
3	Далее будем считать, что все простые числа, которые написаны на доске больше 2. Рассмотрим числа на первой доске. Если вычесть или просуммировать два простых числа, то в результате будет четное число. Так как нечет. $\pm$нечет. $=$ чет. получим, что в любом случае на первой доске будет нечетное число так как чисел 2025. Рассмотрим числа на второй доске. На четные числа мы можем не обращать внимания они не влияют на конечное значение, а число нечетных нечетно значит в итоге получим нечетное число. Отсюда на двух досках в итоге будут написаны два нечетных числа, значит всегда в конце игре будет четное число. Отсюда приходим к выводу, что ни ходы Ивана, ни ходы Иннокентия не влияют на результат игры, а поэтому мы можем привести совершенно любую тактику. Если предположить что на первой доске было написано нечетное число раз число 2, то получим ситуацию чет и нечет для которой нет универсальной тактики.	20
4	 Нарисуем для наглядности граф. С помощью него посчитаем число путей. А именно можно посчитать сколько путей ведут до каждой вершины начиная от 4. Всего путей **24**. (см рис. 1)  Для поиска кратчайшего пути ищем минимальный путь до “5” – 3. Из этого следует, что расстояние до 7 в любом случае будет не меньше 6. Так как до “8” кратчайший путь составляет 6, то значит меньше 6 расстояния не может быть. Значит самый короткий путь **417(длина 6)**.	25
5	А) Программа выведет “нет”. Б) Программа выведет “да”. В) Этот алгоритм проверяет число на простоту. Выводит “да”, если	20

	число составное, а “нет” если простое. Каждую итерацию k сравнивается с number если k меньше number, то мы проверяем делится ли number на k. Причем заметим, что number находится в промежутке в промежутке от 2 до number-1 т.е. Это означает что тем самым мы ищем делитель числа number отличный от единицы и самого числа number. Причем после того, как k становится равным number, и у него не найдено делителя, то выводится “нет”, т.е. если число number простое. Из этих рассуждений делаем вид, что данный алгоритм предназначен для проверки числа на простоту. Т.е. элементарный тест на простоту.	

3 вариант

№	Правильный ответ	Балл
1	Ответ на первый вопрос: например, можно сделать так: $\frac{2*2025+2025*2023}{2025*2025}$. По шагам: 1. Умножим число 2 на 2025 получаем $2 * 2025$ 2. Прибавим к числу 2*2025, 2023 раза число 2025: $2 * 2025 + 2025 * 2023$ 3. Поделим получившиеся число на 2025 дважды. Далее напишем, как можно получить из 1 число 2. Например так $\frac{1*2025+2025}{2025}$ По шагам: 1. Умножим единицу на 2025 2. Прибавим 2025 Поделим на 2025	15
2	Так как стоимость одинакова рассмотрим среднюю скорость. В случае отправки файла размером в 63 Кбайт убедимся, что “Дзета-23” отправит этот файл за 7 минут (63/9=7). Найдем какая средняя скорость отправки файла в случае, если выбрать “Кси-23” $\frac{(1+1+2+4+8+16+31)}{7} = 9$. Отсюда можем сделать вывод что можно выбрать любую компанию так как время отправки будет одно и тоже. Отсюда можно сделать вывод что если файл больше, чем 63 Кбайт, то средняя скорость “Кси-23” будет больше, чем средняя скорость “ Тау-23”, а значит интернет от этой компании выгоднее в случае, если размер файла больше 63 Кбайт.	20
3	Далее будем считать, что все простые числа, которые написаны на доске больше 2. Рассмотрим числа на первой доске. Если вычесть или просуммировать два простых числа, то в результате будет четное число. Так как $\text{нечет.} \pm \text{нечет.} = \text{чет.}$ получим, что в любом случае на первой доске будет нечетное число так как чисел 2021. Рассмотрим числа на второй доске. На четные числа мы можем не	20

	обращать внимания они не влияют на конечное значение, а число нечетных нечетно значит в итоге получим нечетное число. Отсюда на двух досках в итоге будут написаны два нечетных числа, значит всегда в конце игре будет четное число. Отсюда приходим к выводу, что ни ходы Ивана, ни ходы Иннокентия не влияют на результат игры, а поэтому мы можем привести совершенно любую тактику. Если предположить что на первой доске было написано нечетное число раз число 2, то получим ситуацию чет и нечет для которой нет универсальной тактики.	
4	  Нарисуем для наглядности граф. С помощью него посчитаем число путей. А именно можно посчитать сколько путей ведут до каждой вершины начиная от 4. Всего путей **24**. (см рис. 1) Для поиска кратчайшего пути ищем минимальный путь до “5” – 3. Из этого следует, что расстояние до 7 в любом случае будет не меньше 6. Так как до “8” кратчайший путь составляет 6, то значит меньше 6 расстояния не может быть. Значит самый короткий путь **417(длина 6)**.	25
5	А) Программа выведет “да”. Б) Программа выведет “нет”. В) Этот алгоритм проверяет число на простоту. Выводит “да”, если число простое, а “нет” если составное. Каждую итерацию k сравнивается с number если k меньше number, то мы проверяем делится ли number на k. Причем заметим, что number находится в промежутке в промежутке от 2 до number-1 т.е. Это означает что тем самым мы ищем делитель числа number отличный от единицы и самого числа number. Причем после того, как k становится равным number, и у него не найдено делителя, то выводится “да”, т.е. если число number простое. Из этих рассуждений делаем вид, что данный алгоритм предназначен для проверки числа на простоту. Т.е. элементарный тест на простоту.	20

Информатика. 6 класс
Критерии оценивания

Задание 1.
Показано что число 1 можно получить используя приведенные операции – 10 баллов
В ходе решения допущена арифметическая ошибка, которая повлияла на ответ, но в случае исправления получится верный ответ– 10 баллов.
Дан полный обоснованный ответ – 15 баллов.

Задание 2.
Дан верный ответ под пунктами “А” или “Б” без обоснования – 0 баллов
Даны только верные ответы без обоснования под пунктами “А” и “Б” – 2 балла
Дан полный обоснованный и верный ответ на вопрос под пунктом “А” – 10 баллов
Дан полный обоснованный и верный ответ на вопрос под пунктом “Б” – 10 баллов
Дан полный обоснованный ответ под пунктами “А” и “Б” – 20 баллов

Задание 3.
Рассмотрено два случая, когда на доске с простыми числами может быть двойка и когда её может не быть – 6 баллов
Если написано утверждение о том что в случае когда на доске с простыми числами четное число двоек, то универсальной стратегии нет – 10 баллов
Показано что на досках в сумме всегда будет четное число и не рассмотрен случай с двойками – 12 баллов
Доказано, что на первой или второй доске в любом случае получается нечетное число в случае, если двойка отсутствует или цифр 2 четное число – 14 баллов.
Доказано, что на первой и второй доске в любом случае получается нечетное число в случае, если двойка отсутствует или цифр 2 четное число – 18 баллов.
Показана что в любом случае будет ничья если на доске с простыми числами цифра 2 написана четное число раз – 20 баллов

Задание 4.
Дан ответ под пунктами “А” или “Б” без обоснования – 0 баллов
Были даны верные ответы без обоснования под пунктами “А” и “Б” – 3 балла
Верно найден кратчайший путь с частичным обоснованием – 4 балла
Верно найден кратчайший путь с полным обоснованием – 6 балла
Таблица была представлена в виде графа – 8 баллов.
Число путей найдено верно, но ответ не обоснован – 10 баллов.
Число путей найдено верно и обосновано любым способом – 16 баллов.
Получены верные обоснованные ответы как для числа путей, так и для самого короткого пути – 25 баллов.

Задание 5.
Верно, написан вывод программы в случае ввода конкретного значения пункты “А” и “Б” – 8 баллов.
Верно определен вывод программы и какую задачу он решает без обоснования -15

баллов.
Верно написан вывод под одной из букв А или Б и при этом верно определено, что делает программа – 16 баллов
Верно определен вывод программы и какую задачу он решает с обоснованием – 20 баллов.

Информатика. 7 класс
Решения

1 вариант

<table>
<tr><th>№</th><th>Правильный ответ</th><th>Балл</th></tr>
<tr><td>1.</td><td>По всем этим условиям можно построить таблицу
<table>
<tr><td></td><td>В</td><td>Е</td><td>Р</td><td>А</td></tr>
<tr><td>М</td><td>нет</td><td></td><td></td><td>нет</td></tr>
<tr><td>И</td><td>нет</td><td></td><td>нет</td><td></td></tr>
<tr><td>Р</td><td></td><td>нет</td><td></td><td></td></tr>
<tr><td>Ф</td><td></td><td></td><td>нет</td><td></td></tr>
</table>
Далее, все пары, представленные в условии (Варвара-информатик, Родион-физик и так далее) разные и могут либо не совпадать, либо совпадать только по одному человеку. Раз пары Родион-физик, Арсений-математик и математик-физик разные, то и Родион, и Арсений не являются ни математиками, ни физиками. Тогда Родион любит русский язык, Арсений – информатик, Варвара физик, а Елизавета – математик.</td><td>20</td></tr>
<tr><td>2.</td><td>Способов расставить двух солдатиков в ряд 2, трёх солдатиков 6, четырёх 24, пятерых 120. Если один солдатик обязательно правее другого, то число способов уменьшается ровно в два раза в каждом случае. Итого мы видим, что разница между способами расставить 4 и 5 солдатиков ровно 48. Ответ 4.</td><td>18</td></tr>
<tr><td>3.</td><td>Заметим, что есть смысле проверять только системы счисления, начиная с восьмеричной. Первое уравнение преобразуется в $3x = 22$. Только в системах счисления с основанием 8, 11 и 14 число 22 будет делиться на 3 (можно проверить перебором). $4x = 7 + 25$ (тут складывать числа нельзя, ибо система заранее неизвестна) можно переложить в вид $4x = 12 + 2y$, где y система счисления. Тогда решение будет только для y = 14 из тех трёх систем, что у нас есть, и x совпадёт. Ответ: система счисления с основанием 14.</td><td>15</td></tr>
<tr><td>4.</td><td>В троичной система для всех натуральных чисел алгоритм будет работать так – сначала, если последняя цифр 1, то алгоритм сработает 1 раз, получит на конце 0, затем вторым заходом поделит на три и удалит этот ноль (если это не последняя цифра в числе!). Если последняя цифра 2, уйдёт на одну итерацию больше, если 0 – на одну меньше. Итого для всех цифр, кроме самой левой, уйдёт операций I + 1, где I это цифра. В сумме получится, что s будет равно сумме цифр плюс количеству цифр минус 1. Для двузначных чисел (в троичной системе) такой результат недостижим, а для трёхзначных самым маленьким числом, подходящим под условие, будет $222_3 = 26$.</td><td>24</td></tr>
<tr><td>5.</td><td>Заметим, что детей 15 лет и старше во второй смене нет. Остальных разобьём на подкатегории:
1) Мальчики 14 лет и младше на второй смене – 1</td><td>23</td></tr>
</table>

	2) Девочки 14 лет и младше на второй смене – 2 3) Мальчики 14 лет и младше на первой смене – 3 4) Девочки 14 лет и младше на первой смене – 4 5) Мальчики 15 лет и старше на первой смене – 5 6) Девочки 15 лет и старше на первой смене – 6 Первый запрос это 1 и 3 категории, второй это 2 и 4, третий 3, 4, 5 и 6, четвёртый это 5 и 6. Нам надо найти сумму 1 и 2 категорий. Если суммировать первый, второй и четвёртый запрос и отнять третий, мы как раз и получим искомую сумму. Итого 55 + 40 + 70 – 110 = 55.	

2 вариант

№	Правильный ответ	Балл
1.	По всем этим условиям можно построить таблицу |	В	Е	Р	А | | М |	| нет | нет | | | И |	|	| нет | | Р | нет |	| нет | | | Ф |	| нет |	| Далее, все пары, представленные в условии (Елизавета-физик, Родион-математик и так далее) разные и могут либо не совпадать, либо совпадать только по одному человеку. Раз пары Родион-математик, Арсений-информатик и математик-информатик разные, то и Родион, и Арсений не являются ни математиками, ни информатиками. Тогда Родион любит физику, Арсений любит русский, Варвара – математику, а Елизавета – информатику.	20
2.	Способов расставить двух солдатиков в ряд 2, трёх солдатиков 6, четырёх 24, пятерых 120, шестерых 720. Если один солдатик обязательно правее другого, то число способов уменьшается ровно в два раза в каждом случае. Итого мы видим, что разница между способами расставить 4 солдатиков (с изначальным правилом) и 5 солдатиков (без этого правила) ровно 108. Ответ 4.	18
3.	Заметим, что есть смысле проверять только системы счисления, начиная с восьмеричной. Первое уравнение преобразуется в $2x = 31$. Только в нечётных системах счисления число 31 будет делиться на 2 (можно проверить перебором). $3x = 23 + 25$ (тут складывать числа нельзя, ибо система заранее неизвестна) можно переложить в вид $4x = 8 + 4y$, где y система счисления. Тогда решение будет только для y = 7 и y = 13 из нечётных систем, а x совпадёт только для второго ответа. Ответ: система счисления с основанием 13.	15
4.	В четверичной системе счисления для всех натуральных чисел алгоритм будет работать так – сначала, если последняя цифр 1, то алгоритм сработает 1 раз, получит на конце 0, затем вторым заходом поделит на четыре и удалит этот ноль (если это не последняя цифра в числе!). Если последняя цифра 2, уйдёт на одну итерацию больше, если 0 – на одну меньше, если 3, то на одну больше, чем в случае	24

	двойки. Итого для всех цифр, кроме самой левой, уйдёт операций I + 1, где I это цифра. В сумме получится, что s будет равно сумме цифр плюс количеству цифр минус 1. Для двузначных чисел (в четверичной системе) такой результат недостижим, а для трёхзначных самым маленьким числом, подходящим под условие, будет $133_4 = 31$.	
5.	Заметим, что детей 15 лет и старше во второй смене нет. Остальных разобьём на подкатегории: 1) Мальчики 14 лет и младше на второй смене – 1 2) Девочки 14 лет и младше на второй смене – 2 3) Мальчики 14 лет и младше на первой смене – 3 4) Девочки 14 лет и младше на первой смене – 4 5) Мальчики 15 лет и старше на первой смене – 5 6) Девочки 15 лет и старше на первой смене – 6 Первый запрос это 1 и 3 категории, второй это 2 и 4, третий 3, 4, 5 и 6, четвёртый это 5 и 6. Нам надо найти сумму 1 и 2 категорий. Если суммировать первый, второй и четвёртый запрос и отнять третий, мы как раз и получим искомую сумму. Итого 65 + 45 + 40 – 100 = 50.	23

3 вариант

№	Правильный ответ	Балл
1.	По всем этим условиям можно построить таблицу (см. ниже) Далее, все пары, представленные в условии (Варвара-информатик, Родион-физик и так далее) разные и могут либо не совпадать, либо совпадать только по одному человеку. Раз пары Родион-физик, Арсений-математик и математик-физик разные, то и Родион, и Арсений не являются ни математиками, ни физиками. Тогда Родион любит русский язык, Арсений – информатик, Варвара физик, а Елизавета – математик.	20
2.	Способов расставить две книги в ряд 2, три 6, четыре 24, пять 120, шесть 720 (каждый раз предыдущее число умножается на число, равное числу книг). Если одна книга обязательно правее другой, то число способов уменьшается ровно в два раза в каждом случае. Итого мы видим, что разница между способами расставить 4 и 5 книг ровно 48. Ответ 4.	18
3.	Заметим, что есть смысле проверять только системы счисления, начиная с пятеричной. Перепишем неравенства в десятичную систему, приняв у как незвестную, обозначающую систему счисления. Получим $3x < 2y + 1$ и $2x > y + 4$. Умножим первое неравенство на 2, второе на 3. Получим, что $3y + 12 < 6x < 4y + 2$.	15

	В	Е	Р	А
М	нет			нет
И	нет		нет	
Р		нет		
Ф			нет	

	Разница между верхней и нижней границей равна y-10, значит нет смысла перебирать десятичную и все системы счисления меньше 10. Из оставшихся 6 подходит только система счисления с основанием 15. Ответ: 15.	
4.	В пятеричной системе счисления для всех натуральных чисел алгоритм будет работать так – будет прибавлять 1 до тех пор, пока не получит на конце 0, затем поделит на пять и удалит этот ноль (если это не последняя цифра в числе!). Получается, что на каждую цифру в записи расходуется не более 5 итераций. Значит, для двухзначных чисел (в пятеричной системе счисления) смотреть это нет смысла, а для трёхзначных мы можем добиться наибольшего числа итераций числом $101_5=26_{10}$. Получится 14 итераций, а раз нам надо 13, то просто добавим единицу к числу. Самым маленьким числом, подходящим под условие, будет $102_5 = 27$.	24
5.	Заметим, что детей 15 лет и старше во второй смене нет. Остальных разобьём на подкатегории: 1) Мальчики 14 лет и младше на второй смене – 1 2) Девочки 14 лет и младше на второй смене – 2 3) Мальчики 14 лет и младше на первой смене – 3 4) Девочки 14 лет и младше на первой смене – 4 5) Мальчики 15 лет и старше на первой смене – 5 6) Девочки 15 лет и старше на первой смене – 6 Первый запрос это 1 и 3 категории, второй это 2 и 4, третий 3, 4, 5 и 6, четвёртый это 5 и 6. Нам надо найти сумму 1 и 2 категорий. Если суммировать первый, второй и четвёртый запрос и отнять третий, мы как раз и получим искомую сумму. Итого 32 + 48 + 55 – 90 = 45.	23

Информатика. 7 класс
Критерии оценивания

1. Только ответ 5 баллов.
2. Только ответ 5 баллов. Есть правильные мысли (например, проведены рассуждения для 3 и 4 солдатиков) 10 баллов.
3. Только ответ 5 баллов.
4. Только ответ 10 баллов. Решение с небольшими недочётами либо недостаточным обоснованием 20 баллов.
5. Только ответ 10 баллов.

Информатика. 8 класс
Решения

<table>
<tr><th colspan="3">1 вариант</th></tr>
<tr><td>№</td><td>Правильный ответ</td><td>Балл</td></tr>
<tr><td>1.</td><td>Заметим, что пары Варвара-робототехник, Аделия-назаровец и футболист-дудинец это три разные пары, у каждой из них есть совпадение по одному человеку и нет по второму. Аделия не дудинец, значит, либо робототехник, либо футболист. Если первое, то Борис из Назарова и футболист, тогда Варвара шахматист-дудинец, а этого быть не может. Получаем, что Аделия футболист, Борис робототехник, а Варвара шахматист. Варвара и Борис не из Минусинска, раз они туда приехали только в прошлом году, значит, из Минусинска Аделия, а раз Варвара шахматист, она не из Дудинки. Имеем Аделия-футболист-Минусинск, Борис-робототехник-Дудинка, Варвара-шахматист-Назарово.</td><td>20</td></tr>
<tr><td>2.</td><td>Проделав эти операции, учитывая, что сортировка может проводиться по-разному в зависимости от того, в каких пределах X, получаем следующие варианты
<table>
<tr><td>-7</td><td>-1</td><td>9-x</td></tr>
<tr><td>7</td><td>x</td><td>11</td></tr>
<tr><td>-2</td><td>3-x</td><td>4</td></tr>
</table>
<table>
<tr><td>-7</td><td>9-x</td><td>-1</td></tr>
<tr><td>7</td><td>11</td><td>x</td></tr>
<tr><td>-2</td><td>-8</td><td>15-x</td></tr>
</table>
<table>
<tr><td>-7</td><td>-1</td><td>9-x</td></tr>
<tr><td>x</td><td>7</td><td>11</td></tr>
<tr><td>5-x</td><td>-4</td><td>4</td></tr>
</table>
Видим, что подходит только первый вариант, и x = 8.</td><td>14</td></tr>
<tr><td>3.</td><td>20. В ячейку L2 можно поставить =СРЗНАЧ(B2:K2), чтобы посчитать среднее значение. В M2 можно использовать =СЧЁТЕСЛИ(B2:K2; "=2"). В N2 написать =ЕСЛИ(ИЛИ(И(L2>=4,2; M2>=1); И(L2<4,2; M2=0)); 1; 0). Найти сумму столбца N.</td><td>21</td></tr>
<tr><td>4.</td><td>N = int(input())
fl = True
x1, x2, x3 = 0, int(input()), int(input())
if (x2*5 - x3) % 4 == 0: b = (x2*5 - x3)//4
else: fl = False
if (5*b - x2) % 4 == 0: a = (5*b - x2)//4
else: fl = False
for i in range(N-2):</td><td>23</td></tr>
</table>

<table>
<tr><td></td><td>

```
    x1 = x2
    x2 = x3
    x3 = int(input())
    if x3 != 5*x2 – 4*x1:
        fl = False
if fl: print(a, b)
else: print('NO')
```

</td><td></td></tr>
<tr><td>5.</td><td>

```
t, k = map(int, input().split())
s = 0
for i in range(t):
    n, m = map(int, input().split())
    if min(n, m) <= k <= n + m - 1:
        s+=1
print(s)
```

</td><td>22</td></tr>
<tr><td colspan="3">2 вариант</td></tr>
<tr><td>№</td><td>Правильный ответ</td><td>Балл</td></tr>
<tr><td>1.</td><td>Заметим, что пары Варвара-футболист, Аделия-назаровец и шахматист-дудинец это три разные пары, у каждой из них есть совпадение по одному человеку и нет по второму. Аделия не дудинец, значит, либо шахматист, либо футболист. Если первое, то Борис из Назарова и шахматист, тогда Варвара робототехник-дудинец, а этого быть не может. Получаем, что Аделия шахматист, Борис футболист, а Варвара робототехник. Варвара и Борис не из Минусинска, раз они туда приехали только в прошлом году, значит, из Минусинска Аделия, а раз Варвара робототехник, она не из Дудинки. Имеем Аделия-шахматист-Минусинск, Борис-футболист-Дудинка, Варвара-робототехник-Назарово.</td><td>20</td></tr>
<tr><td>2.</td><td>Проделав эти операции, учитывая, что сортировка может проводиться по-разному в зависимости от того, в каких пределах X, получаем следующие варианты
<table>
<tr><td>-7</td><td>-1</td><td>9-x</td></tr>
<tr><td>7</td><td>x</td><td>11</td></tr>
<tr><td>-2</td><td>3-x</td><td>4</td></tr>
</table>
<table>
<tr><td>-7</td><td>9-x</td><td>-1</td></tr>
<tr><td>7</td><td>11</td><td>x</td></tr>
<tr><td>-2</td><td>-8</td><td>15-x</td></tr>
</table>
<table>
<tr><td>-7</td><td>-1</td><td>9-x</td></tr>
<tr><td>x</td><td>7</td><td>11</td></tr>
<tr><td>5-x</td><td>-4</td><td>4</td></tr>
</table>
Видим, что подходит только последний вариант, и x = 5.</td><td>14</td></tr>
<tr><td>3.</td><td>4. В ячейку L2 можно поставить =СРЗНАЧ(B2:K2), чтобы посчитать</td><td>21</td></tr>
</table>

	среднее значение. В M2 можно использовать =СЧЁТЕСЛИ(B2:K2; "=2"). В N2 написать =ЕСЛИ(ИЛИ(И(L2>=4,3; M2>=2); И(L2<4,3; M2<=1)); 1; 0). Найти сумму столбца N.	

4.	N = int(input())
	fl = True
	x1, x2, x3 = 0, int(input()), int(input())
	if (x2*3 - x3) % 2 == 0: b = (x2*3 - x3)//2
	else: fl = False
	if (3*b - x2) % 2 == 0: a = (3*b - x2)//2
	else: fl = False
	for i in range(N-2):
	  x1 = x2
	  x2 = x3
	  x3 = int(input())
	  if x3 != 3*x2 – 2*x1:
	  fl = False
	if fl: print(a, b)
	else: print('NO')
		23

5.	t, k = map(int, input().split())
	s = 0
	for i in range(t):
	  n, m = map(int, input().split())
	  if min(n, m) <= k <= n + m - 1:
	  s+=1
	print(s)
		22

3 вариант

№	Правильный ответ	Балл
1.	Заметим, что пары Варвара-футболист, Аделия-минусинец и шахматист-назаровец это три разные пары, у каждой из них есть совпадение по одному человеку и нет по второму. Аделия не назаровец, значит, либо шахматист, либо футболист. Если первое, то Борис из Минусинска и шахматист, тогда Варвара робототехник-назаровец, а этого быть не может. Получаем, что Аделия шахматист, Борис футболист, а Варвара робототехник. Варвара и Борис не из Дудинки, раз они туда приехали только в прошлом году, значит, из Дудинки Аделия, а раз Варвара робототехник, она не из Назарова. Имеем Аделия-шахматист-Дудинка, Борис-футболист-Назарово, Варвара-робототехник-Минусинск.	20
2.	Проделав эти операции, учитывая, что сортировка может проводиться по-разному в зависимости от того, в каких пределах X, получаем следующие варианты -7 | -1 | 9-x 7 | x | 11 -2 | 3-x | 4	14

	<table><tr><td>-7</td><td>9-x</td><td>-1</td></tr><tr><td>7</td><td>11</td><td>x</td></tr><tr><td>-2</td><td>-8</td><td>15-x</td></tr></table> <table><tr><td>-7</td><td>-1</td><td>9-x</td></tr><tr><td>x</td><td>7</td><td>11</td></tr><tr><td>5-x</td><td>-4</td><td>4</td></tr></table> Видим, что подходит только второй вариант, и x = 12.	
3.	47. В ячейку L2 можно поставить =СРЗНАЧ(B2:K2), чтобы посчитать среднее значение. В M2 можно использовать =СЧЁТЕСЛИ(B2:K2; "=5"). В N2 написать =ЕСЛИ(ИЛИ(И(L2>4,2; M2<=2); И(L2<=4,2; M2>=3)); 1; 0). Найти сумму столбца N.	21

4.	N = int(input())
	fl = True
	x1, x2, x3 = 0, int(input()), int(input())
	if (x2*4 - x3) % 3 == 0: b = (x2*4 - x3)//3
	else: fl = False
	if (4*b - x2) % 3 == 0: a = (4*b - x2)//3
	else: fl = False
	for i in range(N-2):
	  x1 = x2
	  x2 = x3
	  x3 = int(input())
	  if x3 != 4*x2 – 3*x1:
	  fl = False
	if fl: print(a, b)
	else: print('NO')
		23

5.	t, k = map(int, input().split())
	s = 0
	for i in range(t):
	  n, m = map(int, input().split())
	  if min(n, m) <= k <= n + m - 1:
	  s+=1
	print(s)
		22

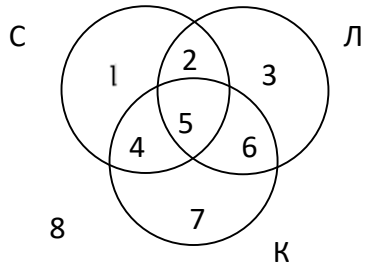
4 вариант		
№	Правильный ответ	Балл
1.	Сумма 170 и 173 не делится на три, а вот 170+175 и 173+175 делятся. Значит, у Аркадия рост не 175. Далее, Аркадий не информатик, а Григорий не физик. При этом, раз пары Аркадий-информатик и Григорий-физик должны совпадать ровно по одному человеку, то рассмотрим два вариант. Пусть Григорий информатик. Тогда физиком становится Денис, Аркадию достанется обществознание. Но	20

<table>
<tr><td></td><td>человек, сдающий обществознание, выше всех остальных, противоречие. Получается, что Аркадий физик, Григорий сдаёт обществозание, а Денис информатику.</td><td></td></tr>
<tr><td>2.</td><td>Проделав эти операции, учитывая, что сортировка может проводиться по-разному в зависимости от того, в каких пределах X, получаем следующие варианты
<table>
<tr><td>13</td><td>10</td><td>23</td></tr>
<tr><td>5</td><td>x</td><td>11</td></tr>
<tr><td>9</td><td>9+x</td><td>8</td></tr>
</table>
<table>
<tr><td>13</td><td>10</td><td>23</td></tr>
<tr><td>5</td><td>11</td><td>x</td></tr>
<tr><td>9</td><td>9+x</td><td>8</td></tr>
</table>
<table>
<tr><td>13</td><td>10</td><td>9+x</td></tr>
<tr><td>5</td><td>11</td><td>x</td></tr>
<tr><td>4</td><td>12</td><td>8-x</td></tr>
</table>
Видим, что подходит только третий вариант, и x = 15.</td><td>14</td></tr>
<tr><td>3.</td><td>13. В ячейку L2 можно поставить =СРЗНАЧ(B2:K2), чтобы посчитать среднее значение. В M2 можно использовать =ABS($L2 - B2) и распространить его на 10 клеток вправо и вниз до упора. В W2 написать =СУММ(M2:V2). Найти максимум столбца W.</td><td>21</td></tr>
<tr><td>4.</td><td>N = int(input())
fl = True
x1, x2, x3 = 0, int(input()), int(input())
if x3 % x2 == 0: b = x3//x2
else: fl = False
if x2 % b == 0: a = x2//b
else: fl = False
for i in range(N-2):
  x1 = x2
  x2 = x3
  x3 = int(input())
  if x3 != x2*x1:
  fl = False
if fl: print(a, b)
else: print('NO')</td><td>23</td></tr>
<tr><td>5.</td><td>t, k = map(int, input().split())
s = 0
for i in range(t):
  n, m = map(int, input().split())
  if min(n, m) <= k <= n + m - 1:
  s+=1
print(s)</td><td>22</td></tr>
</table>

Информатика. 8 класс
Критерии оценивания

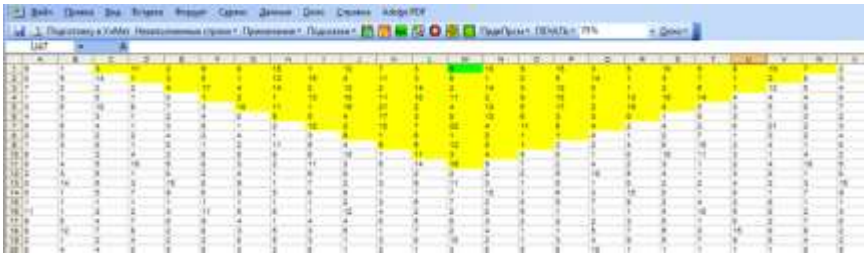
1. Только ответ 5 баллов. За рассуждения без применения того, что пары познакомились в разное время и потому неодинаковы 10 баллов.
2. Только ответ 5 баллов.
3. Если формулы были сделаны лишь наполовину, а потом была сделана не удачная попытка перебора, ставилось 10-15 баллов. За небольшие продвижения 5 баллов.
4. За в целом правильную программу, но без проверок на делимость 15 баллов. За некоторые правильные идеи ставилось 5-10 баллов.
5. За проверку неравенства в одну сторону (но правильную) и во всём остальном правильную программу 15 баллов. За некоторые правильные идеи 5-10 баллов.

Информатика. 9 класс
Решения

<table>
<tr><th colspan="4">1 вариант</th></tr>
<tr><th>№</th><th>Ответ</th><th>Балл</th><th>Решение</th></tr>
<tr><td>1.</td><td>7</td><td>15</td><td>Обозначим основание неизвестной системы счисления за x. Тогда условие задачи можно записать следующим уравнением:
$235_x=65_x+104_x+33_x$.
Как известно, любое число в позиционной системе счисления с основанием x может быть представлено в виде разложения по степеням числа x:
$a = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} +\ldots+ a_2 \cdot x^2 + a_1 \cdot x + a_0 + a_{-1} \cdot x^{-1} +\ldots+ a_{-m} \cdot x^{-m}$,
где x – основание системы счисления, a_i – цифры числа a в x-ичной системе счисления.
Тогда условие задачи перепишется в виде:
$2*x^2+3*x+5=6*x+5+1*x^2+4+3*x+3$
Получаем квадратное уравнение: $x^2-6*x-7=0$
$D=36+28=64=8^2$
У этого уравнения два корня: $x_1=-2$ $x_2=7$.
Так как основанием системы счисления может быть только положительное число, следовательно, x=7, и все числа в условии задачи приведены в семеричной системе счисления.
Ответ: 7.</td></tr>
<tr><td>2.</td><td>76</td><td>20</td><td>Для решения задачи необходимо построить диаграмму Эйлера-Венна. На ней отобразим три множества событий. Множество С – множество школьников, которые умеют кататься на сноуборде, множество Л – на лыжах и множество К – на коньках.
На диаграмме обозначим каждую область цифрой: так, например, область 5 – это С ∩Л ∩ К – те, кто умеет кататься на всех трёх видах спортивного зимнего инвентаря: сноуборде, лыжах и коньках. В других обозначениях можно записать, что 5 – это С И Л И К или С &Л & К.
Область 2 – это (С ∩ Л) ∩⌐ К или (С &Л) & ⌐К – те, кто умеет кататься на сноуборде и лыжах, но не умеет кататься на коньках. Область 1 – (С ∩ ⌐ Л) ∩⌐ К или (С &⌐ Л) & ⌐К – те, кто умеет кататься на сноуборде, но не на лыжах и коньках. И так далее. Последняя область 8 – те, кто не умеют кататься ни на сноуборде, ни на лыжах, ни на коньках.
 

Количество школьников, соответствующих событиям каждой области i, будем обозначать через N_i. По условию задачи нам нужно найти, сколько всего девятиклассников принимало участие в анкетировании – т.е. суммарное значение количества школьников во всех областях: $N_1 + N_2 + N_3 + N_4 + N_5 + N_6 + N_7 + N_8$.
Теперь посмотрим, что нам известно.</td></tr>
</table>

			1) Кататься на сноуборде умеют 26 ребят. Это $\mathbf{N_1 + N_2 + N_4 + N_5 = 26.}$ 2) Кататься на лыжах умеют 38 ребят: $\mathbf{N_2 + N_3 + N_5 + N_6 = 38.}$ 3) Кататься на коньках умеют 47 человек. $\mathbf{N_4 + N_5 + N_6 + N_7 = 47.}$ 4) На лыжах и на коньках умеют кататься 20 школьников: $\mathbf{N_5 + N_6 = 20.}$ 5) Умеют кататься на сноуборде и на коньках — 12 школьников: $\mathbf{N_4 + N_5 = 12.}$ 6) Умеют кататься на сноуборде и на лыжах — 11 школьников: $\mathbf{N_2 + N_5 = 11.}$ 7) Количество школьников, которые не умеют кататься ни на сноуборде, ни на лыжах, ни на коньках, равно 3: $\mathbf{N_8 = 3.}$ 8) Количество школьников, которые умеют кататься на более чем одном спортивном зимнем инвентаре, составило 33 человека. Очевидно, что в данное множество должны входить как школьники, катающиеся на двух инвентарях, так и на трёх: $\mathbf{N_2 + N_4 + N_5 + N_6 = 33.}$ 9) Из пункта 4) мы знаем, что $\mathbf{N_5 + N_6 = 20.}$ Также мы знаем из пунктов 5) и 6), что $\mathbf{N_4 + N_5 = 12}$ **и** $\mathbf{N_2 + N_5 = 11.}$ При сложении этих трех уравнений получим, что $\mathbf{N_2 + N_4 + 3*N_5 + N_6 = 20+12+11=43}$ (1). Из предыдущего пункта нам известно, что $\mathbf{N_2 + N_4 + N_5 + N_6 = 33}$ (2). Вычтем из уравнения (1) уравнение (2) и получим, что $\mathbf{2*N_5 = 43-33=10.}$ Значит, $\mathbf{N_5 = 10/2=5.}$ 10) Так как $\mathbf{N_5 + N_6 = 20}$, а $\mathbf{N_5 = 5,}$ значит, N_6 =20-5=15. 11) $\mathbf{N_4 + N_5 = 12}$. Значит, $\mathbf{N_4 = 12-5=7}$. 12) $\mathbf{N_2 + N_5 = 11}$. Значит, $\mathbf{N_2 = 11-5=6}$. 13) В круге «Сноуборд» находится 26 ребят. Это $\mathbf{N_1 + N_2 + N_4 + N_5 = 26.}$ Мы уже знаем, что $\mathbf{N_2 = 6,}$ $\mathbf{N_4 = 7,}$ **а** $\mathbf{N_5 = 5.}$ Отсюда, $\mathbf{N_1 = 26 – N_2 - N_4 – N_5 = 26 – 6-7-5=8.}$ 14) В круге «Лыжи» находится 38 человек. Это $\mathbf{N_2 + N_3 + N_5 + N_6 = 38.}$ Мы уже знаем, что $\mathbf{N_2 = 6,}$ $\mathbf{N_5 = 5}$, $\mathbf{N_6 = 15.}$ Отсюда, $\mathbf{N_3 = 38 – N_2 - N_5 – N_6 = 38 – 6 – 5-15=12.}$ 15) В круге «Коньки» находится 47 человек. Это $\mathbf{N_4 + N_5 + N_6 + N_7 = 47.}$ Мы уже знаем, что $\mathbf{N_4 = 7,}$ $\mathbf{N_5 = 5}$ и $\mathbf{N_6 = 15.}$ Отсюда, $\mathbf{N_7 = 47 – N_4 - N_5 – N_6 = 47 – 7 – 5-15=20.}$ 16) Нам нужно найти суммарное значение количества школьников во всех областях. В принципе, нам уже всё известно для решения задачи: $\mathbf{N_1 + N_2 + N_3 + N_4 + N_5 + N_6 + N_7 + N_8 = 8+6+12+7+5+15+20+3=76.}$ Итого, всего в анкетировании принимало участие 76 девятиклассников. **Ответ: 76**.

3.	665	10	Из условия задачи следует, что нам нужно найти сумму чисел, хранящихся в ячейках заданного диапазона (причем не всех, а соответствующих некоторому ограничению). В условиях задачи имеются три ограничения: 1) можно отойти от начальной точки сбора не далее 10 клеток (по горизонтали или вертикали). 2) собрать из каждой ячейки и унести с собой не более 12 , т.е. если в ячейке находится число больше 12, то взять из нее мы можем только 12. 3) каждый десятый собранный ресурс (т.е. кристалл) отчисляется в фонд взаимопомощи (т.е. отнимается от собранных кристаллов). Очевидно, что третье ограничение уже является финальным, и его можно вычислить на последнем шаге: после того, как мы посчитаем общее количество собранных кристаллов, отнимем от итоговой суммы каждый десятый кристалл. Теперь обратите внимание на первое ограничение: отойти можно только на 10 ячеек. Для этого варианта точка начала сбора находилась в ячейке M1. Если двигаться по этой строке (1) влево на 10 ячеек, то мы можем пройти до ячейки C1. Если двигаться вправо – до W1. Таким образом, диапазон сбора в первой строке равен: C1:W1. Если мы пойдем во вторую строку, то мы уже один ход потратим, чтобы перейти во вторую строку – мы попадем в ячейку M2. И из 10 ходов нам останется сделать только уже 9. Если пойдем влево, то можем сдвинуться до D2, если пойдем вправо – до V2. Диапазон сбора во второй строке равен D2::V2. Если мы идем в строку № 3, то два хода тратим на передвижение в неё, остается только 8 ходов для перемещения. Получаем диапазон E3:U3. И так далее. В одиннадцатой строке получаем только одну ячейку: M11. На рисунке показан полученный набор ячеек для сбора:  Теперь решим вопрос с суммированием с учётом второго ограничения (что мы можем брать только максимум 12 из каждой ячейки). Для решения этой задачи можно пользоваться двумя способами. **Первый способ (менее эффективный):** 1) Из исходной таблицы получить измененную с помощью формулы =ЕСЛИ(A1>12;12;A1) – это делаем для каждой ячейки. В измененной таблице все значения>12 заменятся на 12. Например, новые значения нашего диапазона находятся в строках с 42 по 81. 2) Далее просто выполняем суммирование по всем одиннадцати строкам найденного диапазона (в измененной таблице). Каждую формулу запишем в отдельной ячейке: =СУММ(C42:W42), =СУММ(D43:V43), =СУММ(E44:U44), =СУММ(F45:T45), =СУММ(G46:S46),

			=СУММ(H47:R47), =СУММ(I48:Q48), =СУММ(J49:P49), =СУММ(K50:O50), =СУММ(L51:N51), =СУММ(M52:M52). 3) После этого просуммируем полученные одиннадцать сумм. Например, так: ==СУММ(O82:O92). В результате вычислений по заданному файлу сумма будет равна **739.** Второй способ суммирования (эффективный): 1) Проходим по каждой строке в найденном диапазоне и считаем сумму значений <=12. =СУММЕСЛИ(C1:W1;"<=12"); =СУММЕСЛИ(D2:V2;"<=12"); =СУММЕСЛИ(E3:U3;"<=12") и т.д. для всех одиннадцати строк. 2) Далее в каждой строке найденного диапазона считаем количество ячеек со значениями >12 так: СЧЁТЕСЛИ(C1:W1;">12"). Найденное количество умножаем на 12. Так делаем для каждой из 11-ти строк. 3) После чего к первой сумме (в которой только числа<=12) прибавляем количество чисел >12 (замененных на 12) и умножаем на 12. Т.е. для каждой строки формула будет выглядеть так: =СУММЕСЛИ(C1:W1;"<=12")+ СЧЁТЕСЛИ(C1:W1;">12")*12 =СУММЕСЛИ(D2:V2;"<=12")+ СЧЁТЕСЛИ(D2:V2;">12")*12 И т.д. 3) После этого суммируем полученные одиннадцать сумм. Например, так: ==СУММ(O42:O52) В результате вычислений по заданному файлу сумма будет равна **739.** **Остался последний шаг:** отнять от полученной суммы каждый десятый собранный ресурс (который отчисляется в фонд взаимопомощи). 1/10=0,1 Т.е. нужно взять от суммы 0,1, и затем отнять: 739-739*0,1. Можно сделать это в одно действие: 739*0,9. Если сумма хранилась в ячейке O53, то формула будет выглядеть так: =O53*0,9 Полученное значение равно 665,1. В ответе должно быть целое число (так как кристаллы не делятся на части). Тогда ответ равен 665. Ответ: максимальное количество кристаллов, которое получит персонаж Бельчонка при всех заданных ограничениях, равно **665**.
4.		25	Из условий задачи следует, что у нас имеется массив mas целых чисел размера N. Нам необходимо найти сумму всех элементов массива mas[1]+mas[2]+..mas[N]=sum. Затем определить такой номер k элемента массива, сумма элементов до которого (включая него самого) mas[1]+mas[2]+..mas[k]<=sum/2 (половине начальной суммы.) Причем mas[1]+..+mas[k]+mas[k+1]>sum/2. Приведем наиболее эффективный алгоритм решения задачи. **Алгоритм 1.** mas – массив элементов sum – сумма всех элементов массива s – промежуточная сумма, i – счётчик (номер элемента). 1. Считываем число N.

			2. Инициализируем переменную sum=0; 3. В цикле for от 1 до N выполняем считывание элементов массива mas[i]. (Если позволяет язык программирования, то сразу же накапливаем сумму элементов массива: sum=sum+mas[i].) 4. Если суммирование не выполнялось при считывании, то выполняем его сейчас. 5. Присваиваем s=sum/2, i=0. 6. Идем в цикле while пока s>0 и выполняем в нём следующие действия: 6.1) от частичной суммы s отнимаем значение текущего элемента массива: s-mas[i]. 6.2) увеличиваем счётчик I, т.е. увеличиваем номер элемента массива Выход из цикла while произойдет после того, как s станет <0, т.е. разница между половиной начальной суммы sum и накопленных значений станет <0. 7. Полученное значение переменной i будет являться номером элемента массива, сумма элементов до которого (включая него самого) ближе всего к sum/2. Выводим i на экран. Так же эффективным будет являтся схожий алгоритм, который ведет накопление частичной суммы и сравнивает ее с половиной начальной (sum/2). **Алгоритм 2.** mas – массив элементов sum – сумма всех элементов массива s – промежуточная сумма, i – счётчик, k – номер искомого элемента (со значением ближе всего к половине sum, с нижней грани). 1. Считываем число N. 2. Инициализируем переменную sum=0; 3. В цикле for от 1 до N выполняем считывание элементов массива mas[i]. (Если позволяет язык программирования, то сразу же накапливаем сумму элементов массива: sum=sum+mas[i].) 4. Если суммирование не выполнялось при считывании, то выполняем его сейчас. 5. Присваиваем s=0, k=0. 6. Идем в цикле for от 1 до N пока sum/2-s>=0 и выполняем в нём следующие действия: 6.1) к частичной сумме s прибавляем значение текущего элемента массива: s=s+mas[i]. 6.2) выполняем проверку: если sum/2-s>=0, то увеличиваем номер элемента массива k; Если это не так, то выполняем выход из цикла. Т.е. 7. Полученное значение переменной k будет являться номером элемента массива, сумма элементов до которого (включая него самого) ближе всего к sum/2. Выводим k на экран. Использование векторов или списков при решении данной задачи является еще более эффективным способом решения задачи. Другие алгоритмы решения данной задачи (с отниманием и возвратом, с

накоплением разниц в других массивах и т.д.) не являются эффективными.

Пример программы на языке C++:

```
#include <iostream>
using namespace std;

int main()
{
    long long n, sum=0,s=0;
    int i;
    int mas[10000];
    cin >> n;
    for (i = 0; i < n; ++i) {
          cin >> mas[i];
          sum =sum+mas[i];
     }
     s =sum/2;
     i = 0;
     while (s >= 0) {
          s=s-mas[i];
          i++;
     }
     i--;
     cout << i;
     return 0;
}
```

Второй алгоритм решения (к тому же с векторами).

Пример программы на языке C++:

```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    long long n, sum=0,s=0;
    int i,k=0;
    cin >> n;
    vector <long long>mas(n);
    for (int i = 0; i < n; i++) {
        cin >> mas[i];
        sum += mas[i];
    }
    s=0;
    for (i = 0; i < n; i++) {
        s+= mas[i];
        if (s<=sum/2) k++;
        else break;//выход из цикла
    }
    cout << k;
    return 0;
}
```

			Пример программы на языке Python:
			`n = int(input())`
			`mas = list(map(int, input().split()))`
			`sm = sum(mas) // 2`
			`s = 0`
			`k = 0`
			`for i in range(len(mas)):`
			`  s += mas[i]`
			`  if s <= sm:`
			`  k += 1`
			`  else:`
			`  break`
			`print(k)`

5.		30	Из условий задачи следует, что программа должна считать массив mas положительных целых чисел размера N (N>2). Затем нужно перебрать все пары различных элементов последовательности и найти количество таких пар, сумма элементов которых является квадратом некоторого целого числа. Перебор возможных пар элементов никак не ускорить. А вот проверку на то, что сумма двух элементов является квадратом можно выполнить двумя способами: неэффективным и эффективным. **1 вариант решения. С полным перебором при проверке на квадрат (неэффективный способ).** mas – массив элементов s – сумма пары элементов, i,j,l – счётчики, k – количество пар, сумма которых является квадратами 1. Считываем число N. 2. В цикле от 1 до N считываем элементы массива (т.е. числа последовательности). 3. Во вложенных циклах перебираем все возможные пары чисел. Первый цикл по i перебирает все элементы от 1 до предпоследнего (i – индекс элемента массива, который будет первым в паре). Второй по j перебирает все элементы от следующего за текущим (т.е. j=i+1) до последнего (j – индекс элемента массива, который будут вторым в паре). 3.1) находим сумму s=mas[i] + mas[j]. 3.2) Выполняем проверку суммы s, что она является квадратом некоторого числа. Для этого в цикле перебираем числа l (от 1 до sqrt(s)), возводим их в квадрат и сравниваем с s. Если l^2=s, то увеличиваем счётчик k на 1. 4) Повторяем действия для следующих пар. 5) В полученном значении переменной k будет содержаться количество пар, сумма которых является квадратом. Выводим значение k на экран. **2 вариант решения. С проверкой на квадрат с помощью функции квадратного корня (эффективный способ).** mas – массив элементов s – сумма пары элементов, i,j – счётчики, k – количество пар, сумма которых является квадратами

1. Считываем число N.
2. В цикле от 1 до N считываем элементы массива (т.е. числа последовательности).
3. Во вложенных циклах перебираем все возможные пары чисел.
Первый цикл по i перебирает все элементы от 1 до предпоследнего (i – индекс элемента массива, который будет первым в паре). Второй по j перебирает все элементы от следующего за текущим (т.е. j=i+1) до последнего (j – индекс элемента массива, который будут вторым в паре).
3.1) находим сумму s=mas[i] + mas[j].
3.2) Выполняем проверку суммы s, что она является квадратом некоторого числа. Извлекаем квадратный корень из числа s с помощью специальной функции языка (sqrt, pow и т.д.). Если найденный квадратный корень из s является целым числом, то s – квадрат некого числа, тогда увеличиваем счётчик k на 1.
4. Повторяем действия для следующих пар.
5. В полученном значении переменной k будет содержаться количество пар, сумма которых является квадратом. Выводим значение k на экран.

Использование векторов или списков при решении данной задачи является еще более эффективным способом решения задачи.

Пример программы на языке C++:

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    long long n, k=0;
    double s;
  int mas[10000];
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> mas[i];
    }
    for (int i = 0; i < (n - 1); i++) {
        for (int j = i + 1; j < n; j++) {
            s=pow(mas[i]+mas[j],1.0/2);
      if(int(s)==s)
         k++;
        }
    }
    cout << k;
    return 0;
}
```

Пример программы на языке C++ (с вектором):

```
#include <iostream>
#include <vector>
#include <cmath>
using namespace std;
```

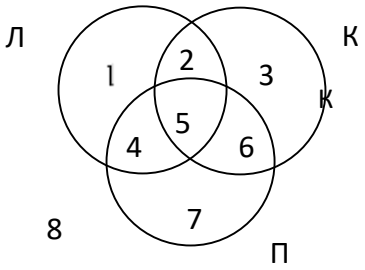
```
int main() {
    long long n, k=0;
    float s;
    cin >> n;
    vector <float>mas(n);

    for (int i = 0; i < n; i++) {
        cin >> mas[i];
    }
    for (int i = 0; i < (n - 1); i++) {
        for (int j = i + 1; j < n; j++) {
            s=pow(mas[i]+mas[j],1.0/2);
      if(int(s)==s)
         k++;
        }
    }
    cout << k;
    return 0;
}
```

Пример программы на языке Python:

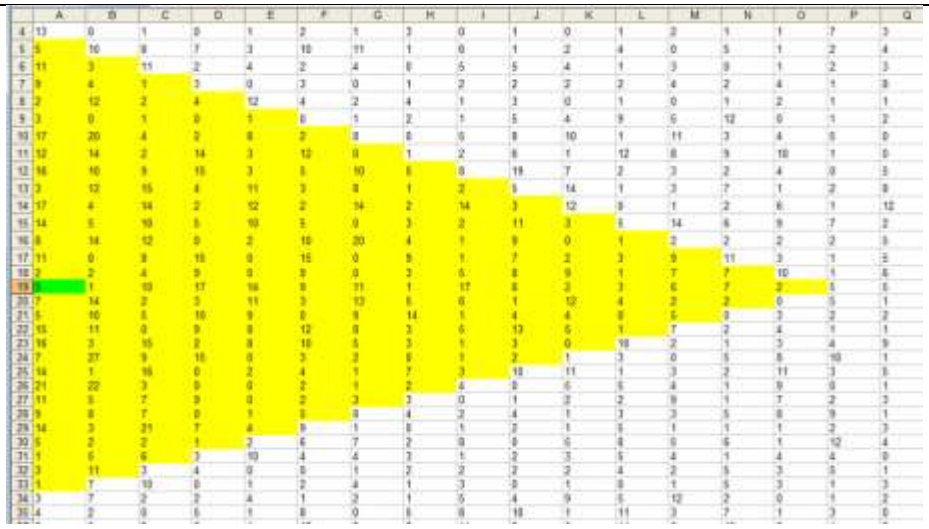
```
n = int(input())
mas = []
k= 0
s=0
for i in range(n):
    mas.append(input())
for i in range(n):
    for j in range(i + 1, n):
       s= int(mas[i]) + int(mas[j])
       if (s) ** 1/2 == round((s) ** 1/2):
          k+=1
print(k)
```

2 вариант

№	Ответ	Балл	Решение
1.	8	15	Обозначим основание неизвестной системы счисления за x. Тогда условие задачи можно записать следующим уравнением: $204_x=43_x+141_x$. Как известно, любое число в позиционной системе счисления с основанием x может быть представлено в виде разложения по степеням числа x: $a = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_2 \cdot x^2 + a_1 \cdot x + a_0 + a_{-1} \cdot x^{-1} + \ldots + a_{-m} \cdot x^{-m}$, где x – основание системы счисления, a_i – цифры числа a в x-ичной системе счисления. Тогда условие задачи перепишется в виде: $2*x^2+0*x+4=4*x+3+1* x^2+4*x+1$ Получаем уравнение: $x^2-8*x=0$ У этого уравнения два корня: $x_1=0$ $x_2=8$. Так как основанием системы счисления может быть только положительное число, следовательно, x=8, и все числа в условии задачи приведены в восьмеричной системе счисления. Ответ: 8

2	56	20	Для решения задачи необходимо построить диаграмму Эйлера-Венна. На ней отобразим три множества событий. Множество Л – множество участников, решивших задачи на тему «логика», множество К – на тему «комбинаторика» и множество П – на тему «программирование». На диаграмме обозначим каждую область цифрой: так, например, область 5 – это **Л ∩ К ∩ П** – те, кто решил все три задачи: на логику, комбинаторику и программирование. В других обозначениях можно записать, что 5 – это Л **И** К **И** П или Л **&**К **&** П. Область 2 – это **(Л ∩ К) ∩⌐ П** или (Л **&**К) **&** ⌐**П** – те, кто решил задачу на логику и комбинаторику, но не решил задачу по программированию. Область 1 – **(Л ∩ ⌐ К) ∩⌐ П** или (Л **&⌐** К) **&** ⌐**П** –те, кто решил только одну задачу на логику, но не решил задачи на комбинаторику и программирование. И так далее. Последняя область 8 – те, кто ничего не решил.  Количество школьников, соответствующих событиям каждой области **i**, будем обозначать через N_i. По условию задачи нам нужно найти количество всех участников олимпиады, т.е. сумму всех областей: $N_1 + N_2 + N_3 + N_4 + N_5 + N_6 + N_7 + N_8$. Теперь посмотрим, что нам известно. 1) С задачей по теме «Логика» справились 26 человек. Это $N_1 + N_2 + N_4 + N_5 = 26$. 3) Задачу по комбинаторике решили 23 участника. $N_2 + N_3 + N_5 + N_6 = 23$. 4) Задачу по программированию решили 27 человек. $N_4 + N_5 + N_6 + N_7 = 27$. 5) Пять участников решили задачи на логику и комбинаторику (причём часть из них еще выполнили задание по программированию): $N_2 + N_5 = 5$. 6) Ещё 10 участников решили задачи по логике и программированию: $N_4 + N_5 = 10$. 7) А 9 человек решили задачи по комбинаторике и программированию: $N_5 + N_6 = 9$. 8) Два участника не смогли решить ни одной задачи: $N_8 = 2$. 9) Количество школьников, решивших только одну задачу по программированию, равно 5: $N_7 = 5$. 10) 11 участников олимпиады смогли решить всего лишь одну задачу по комбинаторике: $N_3 = 11$. 11) По условию задачи в круге «Комбинаторика» находится 23 человека. Это $N_2 + N_3 + N_5 + N_6 = 23$. Мы уже знаем, что $N_3 = 11$. Значит, $N_2 + N_5 + N_6 = 23 - N_3 = 23 - 11 = 12$ (1). Также из пункта 5) мы знаем, $N_2 + N_5 = 5$. Из пункта 7) известно, что $N_5 + N_6 = 9$.

			Сложим, эти два уравнения. Получим: $N_2+2*N_5+ N_6=14$ (2). Вычтем из уравнения (2) уравнение (1). Получим: $N_5 =14-12=2$. 12) Так как $N_2+N_5=5$, а $N_5 =2$, значит, $N_2=5-2=3$. 13) Так как $N_4 +N_5=10$, значит, $N_4 =10-2=8$. 14) Так как $N_5 + N_6=9$, значит, $N_6=9-2=7$. 15) По условию задачи в круге «Логика» находится 26 человек. Это $N_1 + N_2 + N_4 + N_5 =26$. Мы уже знаем, что $N_2 =3, N_4 =8, N_5= 2$. Значит, $N_1 =26- N_2 – N_4 - N_5=26- 3 - 8 – 2=13$. 16) По условию задачи в круге «Программирование» находится 27 человек: $N_4 + N_5 + N_6 + N_7 =27$. Мы уже знаем, что $N_4 =8, N_5= 2, N_6= 7$. Значит, $N_7=27- N_4 - N_5 - N_6 =27-8-2-7 =10$. 17) Итак, мы нашли значения для всех восьми областей: $N_1 =13, N_2 =3, N_3 =11, N_4 =8, N_5 =2, N_6= 7, N_7 =10, N_8 =2$. **Тогда** $N_1 + N_2 + N_3 + N_4 + N_5 + N_6 + N_7 + N_8=13+3+11+8+2+7+10+2 =56$. **Ответ:** количество участников, принявших участие в олимпиаде, равно 56.
3.	1226	10	Из условия задачи следует, что нам нужно найти сумму чисел, хранящихся в ячейках заданного диапазона (причем не всех, а соответствующих некоторому ограничению). В условиях задачи имеются три ограничения: 1) пират может отойти от места высадки максимум на 14 квадратов (по горизонтали или вертикали). 2) он может собрать из каждой ячейки и унести с собой не более не более 12 монет, т.е. если в ячейке находится число больше 12, то взять из нее мы можем только 12. 3) во время сбора через дыру в мешке выпала каждая двадцатая монетка. Очевидно, что третье ограничение уже является финальным, и его можно вычислить на последнем шаге: после того, как мы посчитаем общее количество собранных монет, отнимем от итоговой суммы каждую двадцатую монету. Теперь обратите внимание на первое ограничение: отойти можно только на 14 ячеек. Для этого варианта точка начала сбора находилась в ячейке A19. Если двигаться по этой строке (1) на 14 клеток вверх, то мы можем пройти до ячейки A5. Если двигаться вниз – до A33. Таким образом, диапазон сбора в первом столбце равен: A5:A33. Если мы пойдем во второй столбец, то мы уже один ход потратим, чтобы перейти во второй столбец – мы попадем в ячейку B19. И из 14 ходов нам останется сделать только уже 13. Если пойдем вверх по этому столбцу, то можем сдвинуться до B6, если пойдем вниз – до B32. Диапазон сбора во втором столбце равен B6:B32. Если мы идем в столбец № 3, то два хода тратим на передвижение в него, остается только 12 ходов для перемещения. Получаем диапазон = C7:C31. И так далее. В последнем пятнадцатом столбце получаем только одну ячейку: O19. На рисунке показан полученный набор ячеек для сбора:



Теперь осталось решить вопрос с суммированием с учётом второго ограничения (что мы можем брать только максимум 12 из каждой ячейки).

Для решения этой задачи можно пользоваться двумя способами.

Первый способ (менее эффективный):

1) Из исходной таблицы получить измененную с помощью формулы =ЕСЛИ(A1>12;12;A1) – это делаем для каждой ячейки. В измененной таблице все значения>12 заменятся на 12.

Например, новые значения нашего диапазона находятся в строках с 44 по 82.

2) Далее просто выполняем суммирование по всем одиннадцати столбцам найденного диапазона. Каждую формулу запишем в отдельной ячейке:

=СУММ(A48:A76), СУММ(B47:B75), ==СУММ(C46:C74), …

=СУММ(N61:N63), =СУММ(O62:O62).

3) После этого суммируем полученные одиннадцать сумм. Например так: ==СУММ(A85:A96)

В результате вычислений по заданному файлу сумма будет равна 1291**.**

Второй способ суммирования (эффективный):

1) Проходим по каждому столбцу в найденном диапазоне и считаем сумму значений <=12. =СУММЕСЛИ(A5:A33;"<=12"),

=СУММЕСЛИ(B6:B32;"<=12"),

=СУММЕСЛИ(C7:C31;"<=12")

И т.д. для всех 11ти столбцов.

2) Далее в каждом столбце найденного диапазона считаем количество ячеек со значениями >12 так: СЧЁТЕСЛИ(A5:A33;">12"). Найденное количество умножаем на 12. Так делаем для каждого из 11ти столбцов.

3) После чего к первой сумме (в которой только числа<=12) прибавляем количество чисел >12 замененное на количество чисел 12, умноженное на 12.

Т.е. для каждой строки формула будет выглядеть так:

=СУММЕСЛИ(A5:A33;"<=12")+ СЧЁТЕСЛИ(A5:A33;">12")*12

=СУММЕСЛИ(B6:B32;"<=12")+ СЧЁТЕСЛИ(B6:B32;">12")*12

И т.д.

3) После этого суммируем полученные одиннадцать сумм. Например, они были в ячейках Z3:Z17. Тогда: =СУММ(Z3:Z17)

В результате вычислений по заданному файлу сумма будет равна 1291**.**

			Остался последний шаг: отнять от полученной суммы каждую двадцатую собранную монету. 1/20=0,05 Т.е. нужно взять от суммы 0,05, и затем отнять: 1291-1291*0,05. Можно сделать это в одно действие: 1291*0,95. Если сумма хранилась в ячейке Z18, то формула будет выглядеть так: =Z18*0,95 Полученное значение равно 1226,45. В ответе должно быть целое число (так как монеты не делятся на части). Округлим 1226,45 в меньшую сторону, ответ будет равен 1226. Ответ: 1226
4.		25	Из условий задачи следует, что у нас имеется массив mas вещественных чисел размера N. Нам необходимо найти сумму всех элементов массива mas[1]+mas[2]+..mas[N]=sum. Затем изменить каждый значение каждого элемента массива mas[i], увеличив его на 30%. После этого нужно определить такой номер k элемента измененного массива, сумма элементов до которого (включая него самого) mas[1]+mas[2]+..mas[k] <=sum и ближе всего к начальной сумме sum. Причем mas[1]+..+mas[k]+mas[k+1]>sum. Приведем наиболее эффективный алгоритм решения задачи. **Алгоритм 1.** mas – массив элементов sum – сумма всех элементов массива s – промежуточная сумма, i – счётчик (номер элемента). 1. Считываем число N. 2. Инициализируем переменную sum=0; 3. В цикле for от 1 до N выполняем считывание элементов массива mas[i]. 4. Находим сумму элементов массива. 5. В цикле for от 1 до N выполняем изменение элементов массива mas[i]=mas[i]*1.3. Примечание: если позволяет язык программирования, то все эти три действия (2, 3 и 4) выполняем в одном первоначальном цикле (при считывании элементов массива). 6. Присваиваем s=sum, i=0. 7. Идем в цикле while пока s>0 и выполняем в нём следующие действия: 7.1) от частичной суммы s отнимаем значение текущего элемента массива: s-mas[i]. 7.2) увеличиваем счётчик i, т.е. увеличиваем номер элемента массива Выход из цикла while произойдет после того, как s станет <0, т.е. разница между начальной суммой sum и накопленной суммой станет <0. 8. Полученное значение переменной i будет являться номером элемента измененного массива, сумма элементов до которого (включая него самого) ближе всего к sum. Выводим i на экран. Так же эффективным будет являтся схожий алгоритм, который ведет накопление частичной суммы и сравнивает ее с начальной. **Алгоритм 2.** mas – массив элементов

sum – сумма всех элементов массива
s – промежуточная сумма, i – счётчик, k – номер искомого элемента (со значением ближе всего к половине sum, с нижней грани).
1. Считываем число N.
2. Инициализируем переменную sum=0;
3. В цикле for от 1 до N выполняем считывание элементов массива mas[i].
4. Находим сумму элементов массива sum.
5. В цикле for от 1 до N выполняем изменение элементов массива mas[i]=mas[i]*1.3.
Примечание: если позволяет язык программирования, то все эти три действия (2, 3 и 4) выполняем в одном первоначальном цикле (при считывании элементов массива).
5. Присваиваем s=0, k=0.
6. Идем в цикле for от 1 до N пока sum-s>=0 и выполняем в нём следующие действия:
6.1) к частичной сумме s прибавляем значение текущего элемента массива: s=s+mas[i].
6.2) выполняем проверку: если sum-s>=0, то увеличиваем номер элемента массива k;
Если это не так, то выполняем выход из цикла. Т.е.
7. Полученное значение переменной k будет являться номером элемента массива, сумма элементов до которого (включая него самого) ближе всего к sum. Выводим k на экран.

Использование векторов или списков при решении данной задачи является еще более эффективным способом решения задачи.

Другие алгоритмы решения данной задачи (с отниманием и возвратом, с накоплением разниц в других массивах и т.д.) не являются эффективными.

Пример программы на языке C++:

```
#include <iostream>
using namespace std;

int main()
{
    float sum=0,s=0;
    int i, n;
    float mas[10000];
    cin >> n;
    for (i = 0; i < n; ++i) {
          cin >> mas[i];
          sum =sum+mas[i];
          mas[i]=mas[i]*1.3;
     }
     s =sum;
     i = 0;
     while (s >= 0) {
          s=s-mas[i];
```

```
            i++;
        }
        i--;
        cout << i;
        return 0;
}
```

Второй алгоритм решения (к тому же с векторами).

Пример программы на языке C++:

```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    float sum=0,s=0;
    int i, n, k=0;
    cin >> n;
    vector <float>mas(n);
    for (i = 0; i < n; ++i) {
          cin >> mas[i];
          sum +=mas[i];
          mas[i]*=1.3;
     }
    s=0;
    for (i = 0; i < n; i++) {
        s+= mas[i];
        if (s<=sum) k++;
        else break;//выход из цикла
    }
    cout << k;
    return 0;
}
```

Пример программы на языке Python:

```
n = int(input())
mas = list(map(float, input().split()))
sm = sum(mas)
s = 0
k = 0
for i in range(len(mas)):
    mas[i]*=1.3
    s += mas[i]
    if s <= sm:
        k += 1
    else:
        break
print(k)
```

5.		30	Из условий задачи следует, что программа должна считать массив mas положительных целых чисел размера N (N>2). Затем нужно перебрать все возможные пары элементов последовательности с различными номерами, для

			каждой пары вычислить их сумму и среди этих сумм найти такое наибольшее значение, которое является кубом некоторого целого числа. Перебор возможных пар элементов никак не ускорить. А вот проверку на то, что сумма двух элементов является кубом можно выполнить двумя способами: неэффективным и эффективным. **1 вариант решения. С полным перебором при проверке на куб (неэффективный способ).** mas – массив элементов s – сумма пары элементов, i,j,l – счётчики, max – наибольшее значение среди всех сумм, являющихся кубами 1. Считываем число N. 2. В цикле от 1 до N считываем элементы массива (т.е. числа последовательности). 3. Переменной max=0. 3. Во вложенных циклах перебираем все возможные пары чисел. Первый цикл по i перебирает все элементы от 1 до предпоследнего (i – индекс элемента массива, который будет первым в паре). Второй по j перебирает все элементы от следующего за текущим (т.е. j=i+1) до последнего (j – индекс элемента массива, который будут вторым в паре). 3.1) находим сумму s=mas[i] + mas[j]. 3.2) Выполняем проверку суммы s, что она является кубом некоторого числа. Для этого в цикле перебираем числа l (от 1 до s-1 или pow(s,1/3)), возводим их в куб и сравниваем с s. Если l^3=s, то сравниваем число s с текущим значением переменной max, если найденное число больше, то заносим его в `max`. 4) Повторяем действия для следующих пар. 5) Полученное значение переменной max будет являться самым наибольшим кубом, среди всех парных сумм элементов. Выводим значение переменной max на экран. **2 вариант решения. С проверкой на куб с помощью функции степени (эффективный способ).** mas – массив элементов s – сумма пары элементов, i,j – счётчики, max – наибольшее значение среди всех сумм, являющихся кубами 1. Считываем число N. 2. В цикле от 1 до N считываем элементы массива (т.е. числа последовательности). 3. Переменной max=0. 3. Во вложенных циклах перебираем все возможные пары чисел. Первый цикл по i перебирает все элементы от 1 до предпоследнего (i – индекс элемента массива, который будет первым в паре). Второй по j перебирает все элементы от следующего за текущим (т.е. j=i+1) до последнего (j – индекс элемента массива, который будут вторым в паре). 3.1) находим сумму s=mas[i] + mas[j].

3.2) Выполняем проверку суммы s, что она является кубом некоторого числа. Извлекаем кубический корень (степень 1/3) из числа s с помощью специальной функции языка (pow и т.д.). Если найденный кубический корень из s является целым числом, то s – куб некого числа, тогда сравниваем число s с текущим значением переменной max, если найденное число больше, то заносим его в `max`.
4) Повторяем действия для следующих пар.
5) Полученное значение переменной max будет являться самым наибольшим кубом, среди всех парных сумм элементов. Выводим значение переменной max на экран.

Использование векторов или списков при решении данной задачи является еще более эффективным способом решения задачи.

Пример программы на языке C++:

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    long long n, max=0;
    int s;
    int mas[10000];
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> mas[i];
    }
    for (int i = 0; i < (n - 1); i++) {
        for (int j = i + 1; j < n; j++) {
            s=pow(mas[i]+mas[j],1.0/3);
            if(int(s)==s&&int(s)>max)
                                max=mas[i]+mas[j];
        }
    }
    cout << max;
    return 0;
}
```

Пример программы на языке C++ (с вектором):

```
# #include <iostream>
#include <vector>
#include <cmath>
using namespace std;

int main() {
    long long n, max=0;
    float s;
    cin >> n;
    vector <float>mas(n);

    for (int i = 0; i < n; i++) {
```

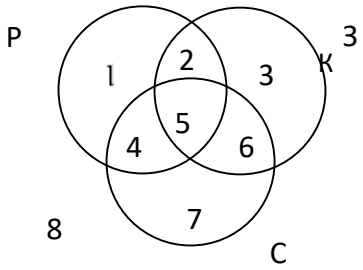
```
        cin >> mas[i];
    }
    for (int i = 0; i < (n - 1); i++) {
        for (int j = i + 1; j < n; j++) {
            s=pow(mas[i]+mas[j],1.0/3);
            if(int(s)==s&&int(s)>max)
                            max=mas[i]+mas[j];

        }
    }
    cout << max;
    return 0;
}
```

Пример программы на языке Python:

```
n = int(input())
mas = []
maxx = 0
s=0
for i in range(n):
    mas.append(input())
for i in range(n):
    for j in range(i + 1, n):
       s= int(mas[i]) + int(mas[j])
       print(s)
       if (s) ** 1/3 == round((s) ** 1/3):
         if s>maxx :
             maxx=s
print(maxx)
```

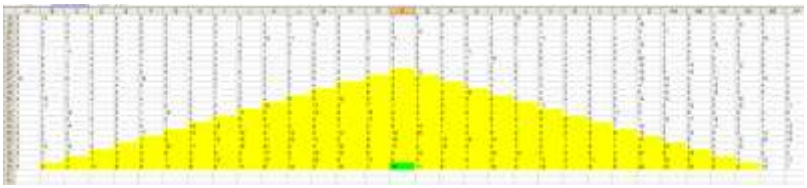
3 вариант

№	Ответ	Балл	Решение
1.	6	15	Обозначим основание неизвестной системы счисления за x. Тогда условие задачи можно записать следующим уравнением: $334_x=151_x+143_x$. Как известно, любое число в позиционной системе счисления с основанием x может быть представлено в виде разложения по степеням числа x: $a = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_2 \cdot x^2 + a_1 \cdot x + a_0 + a_{-1} \cdot x^{-1} + \ldots + a_{-m} \cdot x^{-m}$, где x – основание системы счисления, a_i – цифры числа a в x-ичной системе счисления. Тогда условие задачи перепишется в виде: $3*x^2+3*x+4=1*x^2+5*x+1+1*x^2+4*x+3$ Получаем уравнение: $x^2-6*x=0$ У этого уравнения два корня: $x_1=0$ $x_2=6$. Так как основанием системы счисления может быть только положительное число, следовательно, x=6, и все числа в условии задачи приведены в шестеричной системе счисления. Ответ: 6.

2.	6	20	Для решения задачи необходимо построить диаграмму Эйлера-Венна. На ней отобразим три множества событий. Множество **Р** – множество школьников, которые любят разговаривать на уроках, множество **З** – любят решать задачи и множество **С** – любят спать во время урока. На диаграмме обозначим каждую область цифрой: так, например, область 5 – это **Р** ∩ **З** ∩ **С** – те, кто любят заниматься на уроках всеми тремя делами: разговаривать, решать задачи и спать. В других обозначениях можно записать, что 5 – это Р **И** З **И** С или Р **&З &** С. Область 2 – это **(Р** ∩ **З)** ∩⌐ **С** или (Р **&З**) **&** ⌐С – те, кто любят разговаривать и решать задачи, но не спят на уроках. Область 1 – **(Р** ∩ ⌐ **З)** ∩⌐ **С** или (Р **&**⌐ **З**) **&** ⌐С –те, кто любят только разговаривать, но не любят ни решать задачи, ни спать на уроках. И так далее. Последняя область 8 – те, кто ничего не любят.  Количество школьников, соответствующих событиям каждой области **i**, будем обозначать через N_i. По условию задачи нам нужно найти количество школьников, которые вообще ничего не любят, т.е. N_8. Теперь посмотрим, что нам известно. 1) Всего в классе учатся 38 школьников. Т.е. $N_1 + N_2 + N_3 + N_4 + N_5 + N_6 + N_7 + N_8 = 38$. 2) Любят разговаривать на уроках с кем-то из своих одноклассников 22 человека. Это $N_1 + N_2 + N_4 + N_5 = 22$. 3) 16 школьников любят решать задачи: $N_2 + N_3 + N_5 + N_6 = 16$. 4) Любят поспать во время урока 10 человек: $N_4 + N_5 + N_6 + N_7 = 10$. 5) Среди тех, кто разговаривают на занятиях, постоянно засыпают на уроках 7 человек: $N_4 + N_5 = 7$. 6) Среди тех, кто решают задачи, засыпают только 3 человека: $N_5 + N_6 = 3$. 7) 8 человек успешно совмещают любовь к разговорам и решению задач: $N_2 + N_5 = 8$. 8) Количество школьников, которые любят заниматься на уроках двумя и более делами, равно 14. Очевидно, что в данное множество должны входить как школьники, любящие два занятия, так и те, кто любят все три: $N_2 + N_4 + N_5 + N_6 = 14$. 9) Из пункта 5) мы знаем, что $N_4 + N_5 = 7$. Также мы знаем из пунктов 6) и 7), что $N_5 + N_6 = 3$ и $N_2 + N_5 = 8$. При сложении этих трех уравнений получим, что $N_2 + N_4 + 3*N_5 + N_6 = 7+3+8 = 18$ (1). Из предыдущего пункта нам известно, что $N_2 + N_4 + N_5 + N_6 = 14$ (2). Вычтем из уравнения (1) уравнение (2) и получим, что $2*N_5 = 18-14 = 4$. Значит, $N_5 = 4/2 = 2$.

			10) Так как **N_4 +N_5=7**, а **N_5 = 2,** значит, **N_4 =7-2=5**. 11) **N_5 + N_6 =3**. Значит, **N_6 =3-2=1**. 12) **N_2 + N_5=8**. Значит, **N_2 =8-2=6**. 13) В круге «Разговаривать» находится 22 школьника. Это **N_1 + N_2 + N_4 + N_5 =22.** Мы уже знаем, что **N_2 =6, N_4 =5, N_5 =2.** Отсюда, **N_1 =22 – N_2 - N_4 – N_5 =22 – 6-5-2=9.** 14) В круге **З** («Решать **з**адачи») находится 16 человек: **N_2 + N_3 + N_5 + N_6 =16.** Мы уже знаем, что **N_2 =6, N_5 =2**, **N_6 = 1.** Отсюда, **N_3 =16 – N_2 - N_5 – N_6 =16 – 6 –2-1=7.** 15) В круге **С** («Спать на уроке») находится 10 человек: **N_4 + N_5 + N_6 + N_7 =10.** Мы уже знаем, что **N_4 =5, N_5 =2**, **N_6 = 1.** Отсюда, **N_7 =10 – N_4 - N_5 – N_6 =10 –5 – 2-1=2.** 16) В принципе, нам уже всё известно для решения нашей задачи. Всего количество школьников равно 38: т.е. **N_1 + N_2 + N_3 + N_4 + N_5 + N_6 + N_7 + N_8 =38.** Значит, **N_8 =38 – (N_1 + N_2 + N_3 + N_4 + N_5 + N_6 + N_7).** Напомним значения в этих областях: **N_1 =9, N_2 =6, N_3 =7, N_4 =5, N_5 =2, N_6= 1, N_7 =2.** **Отсюда =38 – (9+ 6 + 7 + 5 + 2 + 1 + 2)= 38-32=6.** **Ответ:** количество школьников, которые вообще ничего не любят, равно 6.
3.	1077	10	Из условия задачи следует, что нам нужно найти сумму чисел, хранящихся в ячейках заданного диапазона (причем не всех, а соответствующих некоторому ограничению). В условиях задачи имеются три ограничения: 1) зонд может отойти от места высадки не далее 14 клеток (по горизонтали или вертикали). 2) он может обследовать не более 10 образцов с каждой клетки местности, т.е. если в ячейке находится число больше 10, то взять из нее мы можем только 10. 3) каждый 10 образец, собранный зондом, был бракованным; бракованные зонды необходимо выбросить. Очевидно, что третье ограничение уже является финальным, и его можно вычислить на последнем шаге: после того, как мы посчитаем общее количество собранных образцов, отнимем от итоговой суммы каждый десятый образец. Теперь обратите внимание на первое ограничение: отойти можно только на 14 ячеек. Для этого варианта точка начала сбора находилась в ячейке P40. Если двигаться по этой строке (№40) влево, то мы можем пройти до ячейки B40. Если двигаться вправо – до AD40. Таким образом, диапазон сбора в строке № 40 равен: B40:AD40. Если мы пойдем в предыдущую строку (№39), то мы уже один ход потратим, чтобы перейти в эту строку – мы попадем в ячейку P39. И из 14 ходов нам останется сделать только уже 13. Если пойдем влево, то можем сдвинуться до C39, если пойдем вправо – до AC39. Диапазон сбора во второй строке равен C39:AC39. Если мы идем в строку № 38, то два хода тратим на передвижение в неё,

остается только 12 ходов для перемещения. Получаем диапазон D38:AB38. И так далее. В последней строке (№26) получаем только одну ячейку: P26.
На рисунке показан полученный набор ячеек для сбора:



Теперь осталось решить вопрос с суммированием с учётом второго ограничения (что мы можем брать только максимум 10 из каждой ячейки).
Для решения этой задачи можно пользоваться двумя способами.

Первый способ (менее эффективный):

1) Из исходной таблицы получить измененную с помощью формулы =ЕСЛИ(A1>10;10;A1) – это делаем для каждой ячейки. В измененной таблице все значения>10 заменятся на 10.
Например, новые значения нашего диапазона находятся в строках с 44 по 82.
2) Далее просто выполняем суммирование по всем пятнадцати строкам найденного диапазона. Каждую формулу записываем в отдельной ячейке: =СУММ(P69:P69), =СУММ(O70:Q70), =СУММ(N71:R71), … =СУММ(M72:S72), … =СУММ(C81:AC81), =СУММ(B82:AD82). Пусть эти суммы располагались в ячейках L84:L98.
3) После этого суммируем полученные 15 сумм. Например, так: =СУММ(L84:L98).
В результате вычислений по заданному файлу сумма будет равна 1197**.**

Второй способ суммирования (эффективный):

1) Проходим по каждой строке в найденном диапазоне и считаем сумму значений <=10 для всех пятнадцати строк:
.=СУММЕСЛИ(P26:P26;"<=10"),
=СУММЕСЛИ(O27:Q27;"<=10"),
=СУММЕСЛИ(M29:S29;"<=10"),
…
=СУММЕСЛИ(C39:AC39;"<=10")б
=СУММЕСЛИ(B40:AD40;"<=10").
2) Далее в каждой строке найденного диапазона считаем количество ячеек со значениями >10 так: СЧЁТЕСЛИ(B40:AD40;">10"). Найденное количество умножаем на 10. Так делаем для каждой из пятнадцати строк.
3) После чего к первой сумме (в которой только числа<=10) прибавляем количество чисел >10 замененное на количество десяток*10.
Т.е. для каждой строки формула будет выглядеть так:
=СУММЕСЛИ(P26:P26;"<=10")+ СЧЁТЕСЛИ(P26:P26;">10")*10
=СУММЕСЛИ(O27:Q27;"<=10")+ СЧЁТЕСЛИ(O27:Q27;">10")*10
И т.д. Пусть эти значения располагались в ячейках P43:P57.
3) После этого суммируем полученные пятнадцать сумм. Вот так: =СУММ(P43:P57)
В результате вычислений по заданному файлу сумма будет равна 1197**.**

			Остался последний шаг: отнять от полученной суммы каждый десятый (бракованный) образец. 1/10=0,1 Т.е. нужно взять от суммы 0,1, и затем отнять: 1197-1197*0,1. Можно сделать это в одно действие: 1197*0,9. Если сумма хранилась в ячейке P58, то формула будет выглядеть так: =P58*0,9 Полученное значение равно 1077,3. В ответе должно быть целое число (так как образцы не делятся на части). Округлим число в меньшую сторону, ответ будет равен 1077. Ответ: 1077
4.		25	Из условий задачи следует, что у нас имеется массив mas целых чисел размера N. Нам необходимо найти сумму всех элементов массива mas[1]+mas[2]+..mas[N]=sum. Затем определить такой номер k элемента массива, сумма элементов до которого (включая него самого) mas[1]+mas[2]+..mas[k]<=sum/2 (половине начальной суммы.) Причем mas[1]+..+mas[k]+mas[k+1]>sum/2. Приведем наиболее эффективный алгоритм решения задачи. **Алгоритм 1.** mas – массив элементов sum – сумма всех элементов массива s – промежуточная сумма, i – счётчик (номер элемента). 1. Считываем число N. 2. Инициализируем переменную sum=0; 3. В цикле for от 1 до N выполняем считывание элементов массива mas[i]. (Если позволяет язык программирования, то сразу же накапливаем сумму элементов массива: sum=sum+mas[i].) 4. Если суммирование не выполнялось при считывании, то выполняем его сейчас. 5. Присваиваем s=sum/2, i=0. 6. Идем в цикле while пока s>0 и выполняем в нём следующие действия: 6.1) от частичной суммы s отнимаем значение текущего элемента массива: s-mas[i]. 6.2) увеличиваем счётчик I, т.е. увеличиваем номер элемента массива Выход из цикла while произойдет после того, как s станет <0, т.е. разница между половиной начальной суммы sum и накопленных значений станет <0. 7. Полученное значение переменной i будет являться номером элемента массива, сумма элементов до которого (включая него самого) ближе всего к sum/2. Выводим i на экран. Так же эффективным будет являтся схожий алгоритм, который ведет накопление частичной суммы и сравнивает ее с половиной начальной (sum/2). **Алгоритм 2.** mas – массив элементов sum – сумма всех элементов массива s – промежуточная сумма, i – счётчик, k – номер искомого элемента (со значением ближе всего к половине sum, с нижней грани).

1. Считываем число N.
2. Инициализируем переменную sum=0;
3. В цикле for от 1 до N выполняем считывание элементов массива mas[i]. (Если позволяет язык программирования, то сразу же накапливаем сумму элементов массива: sum=sum+mas[i].)
4. Если суммирование не выполнялось при считывании, то выполняем его сейчас.
5. Присваиваем s=0, k=0.
6. Идем в цикле for от 1 до N пока sum/2-s>=0 и выполняем в нём следующие действия:
6.1) к частичной сумме s прибавляем значение текущего элемента массива: s=s+mas[i].
6.2) выполняем проверку: если sum/2-s>=0, то увеличиваем номер элемента массива k;
Если это не так, то выполняем выход из цикла. Т.е.
7. Полученное значение переменной k будет являться номером элемента массива, сумма элементов до которого (включая него самого) ближе всего к sum/2. Выводим k на экран.

Использование векторов или списков при решении данной задачи является еще более эффективным способом решения задачи.

Другие алгоритмы решения данной задачи (с отниманием и возвратом, с накоплением разниц в других массивах и т.д.) не являются эффективными.

Пример программы на языке C++:

```
#include <iostream>
using namespace std;

int main()
{
    long long n, sum=0,s=0;
    int i;
    int mas[10000];
    cin >> n;
    for (i = 0; i < n; ++i) {
          cin >> mas[i];
          sum =sum+mas[i];
     }
     s =sum/2;
     i = 0;
     while (s >= 0) {
          s=s-mas[i];
          i++;
     }
     i--;
     cout << i;
     return 0;
}
```

Второй алгоритм решения (к тому же с векторами).

Пример программы на языке C++:

```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    long long n, sum=0,s=0;
    int i,k=0;
    cin >> n;
    vector <long long>mas(n);
    for (int i = 0; i < n; i++) {
        cin >> mas[i];
        sum += mas[i];
    }
    s=0;
    for (i = 0; i < n; i++) {
        s+= mas[i];
        if (s<=sum/2) k++;
        else break;//выход из цикла
    }
    cout << k;
    return 0;
}
```

Пример программы на языке Python:

```
n = int(input())
mas = list(map(int, input().split()))
sm = sum(mas) // 2
s = 0
k = 0
for i in range(len(mas)):
    s += mas[i]
    if s <= sm:
        k += 1
    else:
        break
print(k)
```

5.		30	Из условий задачи следует, что программа должна считать массив mas положительных целых чисел размера N (N>2). Нужно перебрать все пары различных элементов последовательности, найти среди разностей этих пар такие, которые являются квадратом некоторого целого числа, а затем найти среди таких разностей пар наибольшее значение. Перебор возможных пар элементов никак не ускорить. А вот проверку на то, что разность двух элементов является квадратом можно выполнить двумя способами: неэффективным и эффективным. **1 вариант решения. С полным перебором при проверке на квадрат (неэффективный способ).**

			mas – массив элементов s – сумма пары элементов, i,j,l – счётчики, max – max – наибольшее значение среди всех сумм, являющихся квадратами 1. Считываем число N. 2. В цикле от 1 до N считываем элементы массива (т.е. числа последовательности). 3. Во вложенных циклах перебираем все возможные пары чисел. Первый цикл по i перебирает все элементы от 1 до предпоследнего (i – индекс элемента массива, который будет первым в паре). Второй по j перебирает все элементы от следующего за текущим (т.е. j=i+1) до последнего (j – индекс элемента массива, который будут вторым в паре). 3.1) находим разность по модулю s=abs(mas[i] - mas[j]). 3.2) Выполняем проверку разности s, что она является квадратом некоторого числа. Для этого в цикле перебираем числа l (от 1 до sqrt(s)), возводим их в квадрат и сравниваем с s. Если l^2=s, то сравниваем число s с текущим значением переменной max, если найденное число больше, то заносим его в `max`. 4) Повторяем действия для следующих пар. 5) Полученное значение переменной max будет являться самым наибольшим квадратом, среди всех разностей пар элементов. Выводим значение переменной max на экран. **2 вариант решения. С проверкой на квадрат с помощью функции квадратного корня (эффективный способ).** mas – массив элементов s – сумма пары элементов, i,j,l – счётчики, max – max – наибольшее значение среди всех сумм, являющихся квадратами 1. Считываем число N. 2. В цикле от 1 до N считываем элементы массива (т.е. числа последовательности). 3. Во вложенных циклах перебираем все возможные пары чисел. Первый цикл по i перебирает все элементы от 1 до предпоследнего (i – индекс элемента массива, который будет первым в паре). Второй по j перебирает все элементы от следующего за текущим (т.е. j=i+1) до последнего (j – индекс элемента массива, который будут вторым в паре). 3.1) находим разность по модулю s=abs(mas[i] - mas[j]). 3.2) Выполняем проверку суммы s, что она является квадратом некоторого числа. Извлекаем квадратный корень из числа s с помощью специальной функции языка (sqrt, pow и т.д.). Если найденный квадратный корень из s является целым числом, то s – квадрат некого числа, тогда сравниваем число s с текущим значением переменной max, если найденное число больше, то заносим его в `max`. 4) Повторяем действия для следующих пар.

5) Полученное значение переменной max будет являться самым наибольшим квадратом, среди всех разностей пар элементов. Выводим значение переменной max на экран.

Пример программы на языке C++:

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    long long n, max=0;
    double s;
    int mas[10000];
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> mas[i];
    }
    for (int i = 0; i < (n - 1); i++) {
        for (int j = i + 1; j < n; j++) {
            s=pow(abs(mas[i]-mas[j]),1.0/2);
            if(int(s)==s&&int(s)>max)
                    max=abs(mas[i]-mas[j]);
        }
    }
    cout << max;
    return 0;
}
```

Пример программы на языке C++ (с вектором):

```
#include <iostream>
#include <cmath>
#include <vector>
using namespace std;

int main() {
    long long n, max=0;
    double s;
    cin >> n;
    vector <float>mas(n);

    for (int i = 0; i < n; i++) {
        cin >> mas[i];
    }
    for (int i = 0; i < (n - 1); i++) {
        for (int j = i + 1; j < n; j++) {
            s=pow(abs(mas[i]-mas[j]),1.0/2);
            if(int(s)==s&&int(s)>max)
                    max=abs(mas[i]-mas[j]);
        }
    }
    cout << max;
    return 0;
```

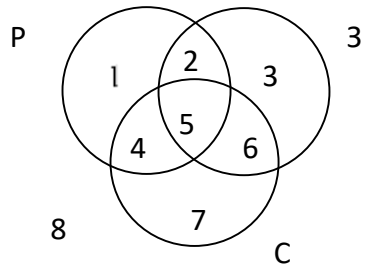
```
}
```

Пример программы на языке Python:

```
n = int(input())
mas = []
maxx = 0
s=0
for i in range(n):
    mas.append(input())
for i in range(n):
    for j in range(i + 1, n):
       s= int(mas[i]) + int(mas[j])
       if (s) ** 1/2 == round((s) ** 1/2):
         if s>maxx :
             maxx=s
print(maxx)
```

4 вариант

№	Ответ	Балл	Решение
1.	8	15	Обозначим основание неизвестной системы счисления за x. Тогда условие задачи можно записать следующим уравнением: $104_x=16_x+13x+30x+23_x$. Как известно, любое число в позиционной системе счисления с основанием x может быть представлено в виде разложения по степеням числа x: $a = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} +\ldots+ a_2 \cdot x^2 + a_1 \cdot x + a_0 + a_{-1} \cdot x^{-1} +\ldots+ a_{-m} \cdot x^{-m}$, где x – основание системы счисления, a_i – цифры числа a в x-ичной системе счисления. Тогда условие задачи перепишется в виде: $1*x^2+0*x+4=1*x+6+1*x+3+3*x+0+2*x+3$ Получаем уравнение: $x^2-7*x-8=0$ $D=49+4*8=81=9^2$ У этого уравнения два корня: $x_1=-1$ $x_2=8$. Так как основанием системы счисления может быть только положительное число, следовательно, x=8, и все числа в условии задачи приведены в восьмеричной системе счисления. Ответ: 8.
2.	12	20	Для решения задачи необходимо построить диаграмму Эйлера-Венна. На ней отобразим три множества событий. Множество **Р** – множество школьников, которые любят разговаривать на уроках, множество **З** – любят решать задачи и множество **С** – любят спать во время урока. На диаграмме обозначим каждую область цифрой: так, например, область 5 – это **Р** ∩ **З** ∩ **С** – те, кто любят заниматься на уроках всеми тремя делами: разговаривать, решать задачи и спать. В других обозначениях можно записать, что 5 – это Р **И** З **И** С или Р **&З &** С. Область 2 – это **(Р** ∩ **З)** ∩⌐ **С** или (Р **&З**) **&** ⌐С – те, кто любят разговаривать и решать задачи, но не спят на уроках. Область 1 – **(Р** ∩ ⌐ **З)** ∩⌐ **С** или (Р **&**⌐ **З) &** ⌐С –те, кто любят только разговаривать, но не любят ни решать задачи, ни спать на уроках. И так далее. Последняя область 8 – те, кто ничего не любят.



Количество школьников, соответствующих событиям каждой области **i**, будем обозначать через N_i. По условию задачи нам нужно найти количество школьников, которые любят заниматься на уроках ровно двумя делами, т.е. $N_2 + N_4 + N_6$.

Теперь посмотрим, что нам известно.

1) Всего в классе учатся 40 школьников. Т.е. $N_1 + N_2 + N_3 + N_4 + N_5 + N_6 + N_7 + N_8 = 40$.

2) Любят разговаривать на уроках с кем-то из своих одноклассников 24 человека. Это $N_1 + N_2 + N_4 + N_5 = 24$.

3) 19 школьников любят решать задачи: $N_2 + N_3 + N_5 + N_6 = 19$.

4) Любят поспать во время урока 12 человек: $N_4 + N_5 + N_6 + N_7 = 12$.

5) Среди тех, кто разговаривают на занятиях, постоянно засыпают на уроках 7 человек: $N_4 + N_5 = 7$.

6) Среди тех, кто решают задачи, засыпают только 4 человека: $N_5 + N_6 = 4$.

7) Десять человек успешно совмещают любовь к разговорам и решению задач: $N_2 + N_5 = 10$.

8) Количество школьников, которые вообще ничего не любят, равно 3: $N_8 = 3$.

9) 11 школьников не любят разговаривать на уроках или спать. Очевидно, что в данное множество должны входить две области: школьники, которые любят только решать задачи (и больше ничего), и те школьники, которые вообще ничего не любят: $N_3 + N_8 = 11$. Из предыдущего пункта нам известно, что $N_8 = 3$. Значит, $N_3 = 11 - N_8 = 11 - 3 = 8$.

10) В круге **З** («**З**адачи») находится 19 человек: $N_2 + N_3 + N_5 + N_6 = 19$. Мы уже знаем, что $N_3 = 8$. Отсюда, $N_2 + N_5 + N_6 = 19 - N_3 = 19 - 8 = 11$ **(1)**.

Из пункта 6) нам известно, что $N_5 + N_6 = 4$. А из пункта **7)** $N_2 + N_5 = 10$. Если мы сложим два этих уравнения, то получим, что $N_2 + 2*N_5 + N_6 = 14$ **(2)**. Вычтем из уравнения (1) уравнение (2). Получим, что $N_5 = 14 - 11 = 3$.

11) Так как $N_5 + N_6 = 4$, то $N_6 = 4 - 3 = 1$.

12) Так как $N_2 + N_5 = 10$, то $N_2 = 10 - 3 = 7$.

13) В пункте 5) сказано, что $N_4 + N_5 = 7$. Значит, $N_4 = 7 - 3 = 4$.

14) В принципе, нам уже всё известно для решения нашей задачи. Нам нужно найти количество школьников, которые любят заниматься на уроках ровно двумя делами, т.е. $N_2 + N_4 + N_6$

$N_2 = 7$, $N_4 = 4$, $N_6 = 1$. Значит, $N_2 + N_4 + N_6 = 7 + 4 + 1 = 12$.

Ответ: количество школьников, которые любят заниматься на уроках ровно двумя делами, равно 12.

3.	123	10	По условию задачи требуется найти максимальное количество моркови, которое может собрать Зайчик при заданных ограничениях, т.е. при движении из заданной точки в одном из четырех направлений (вверх, вниз, влево или вправо). Двигаться назад и менять направление движения нельзя. Так как начальная точка находится внизу поля (ячейка M18), мы можем выбрать только 3 направления: влево, вправо или вверх. Кроме того, есть начальное условие для изменения значений в ячейках: кроты похитили по одной морковке из всех тех лунок на маршруте Зайчика, в которых росло четное количество морковок (если, конечно, там были морковки). Для решения этой задачи можно пользоваться двумя способами. 1) Вначале отнимем морковки из ячеек с четными значениями (если значения >0). Из исходной таблицы получим измененную с помощью формулы = ЕСЛИ(“<=0”;0;ЕСЛИ(ОСТАТ(A1;2)=0;A1-1;A1)) – это делаем для каждой ячейки. Т.е. сразу два условия проверяем, и значение 1 отнимется только у четных чисел >0. Например, новые значения нашего диапазона находятся в строках с 27 по 44. 2) Для каждого из трёх направлений: влево, вправо или вверх от M44 (значение ячейки, соответствующее исходной ячейке M18). Далее просто находим сумму значений ячеек в диапазонах M27:M44, A44:M44 и M44:Y44. =СУММ(M27:M44), =СУММ(A44:M44), =СУММ(M44:Y44). Затем находим максимальное значение среди этих трех значений. Итоговая формула будет выглядеть следующим образом: =МАКС(L46:L48). В результате вычислений по заданному файлу сумма будет равна **123.** Второй способ решения (эффективный): 1) Так как мы отнимаем от каждого чётного числа по единичке, то значит, число похищенных морковок, которое мы должны отнять, совпадает с количеством четных значений морковок. Так что определить собранное количество морковок (в нужном направлении) можно так: посчитать общее количество и отнять количество четных чисел в ячейках этого направления. Для каждого из трёх направлений: влево, вправо или вверх от M18 находим сумму значений ячеек в диапазонах M1:M18, A18:M18 и M18:Y18. =СУММ(M1:M18), =СУММ(A18:M18), =СУММ(M18:Y18). Пусть суммы находятся в ячейках L21, L22, L23. Теперь находим количество чётных значений: =СУММПРОИЗВ(ЕЧИСЛО(M1:M18)*(ОСТАТ(M1:M18;2)=0)), =СУММПРОИЗВ(ЕЧИСЛО(A18:M18)*(ОСТАТ(A18:M18;2)=0)), =СУММПРОИЗВ(ЕЧИСЛО(M18:Y18)*(ОСТАТ(M18:Y18;2)=0)). Пусть количества четных находятся в ячейках M21, M22, M23. Отнимаем количества от соответствующих сумм:

			=L21-M21, =L22-M22, =L23-M23. Эти разницы находятся в ячейках N21, N22, N23. Затем находим максимальное значение среди этих трех значений: =МАКС(N21:N23) В результате вычислений максимальное значение будет равно **123**.
4.	программа	25	Из условий задачи следует, что у нас имеется массив mas вещественных чисел размера N. Нам необходимо найти сумму всех элементов массива mas[1]+mas[2]+..mas[N]=sum. Затем изменить значение каждого второго элемента массива mas[i], увеличив его на 40%. После этого нужно определить такой номер k элемента измененного массива, сумма элементов до которого (включая него самого) mas[1]+mas[2]+..mas[k] <=sum и ближе всего к начальной сумме sum. Причем mas[1]+..+mas[k]+mas[k+1]>sum. Приведем наиболее эффективный алгоритм решения задачи. **Алгоритм 1.** mas – массив элементов sum – сумма всех элементов массива s – промежуточная сумма, i – счётчик (номер элемента). 1. Считываем число N. 2. Инициализируем переменную sum=0; 3. В цикле for от 1 до N выполняем считывание элементов массива mas[i]. 4. Находим сумму элементов массива. 5. В цикле for от 1 до N выполняем изменение элементов массива с четными номерами (i mod 2=0) mas[i]=mas[i]*1.4. Примечание: если позволяет язык программирования, то все эти три действия (2, 3 и 4) выполняем в одном первоначальном цикле (при считывании элементов массива). 6. Присваиваем s=sum, i=0. 7. Идем в цикле while пока s>0 и выполняем в нём следующие действия: 7.1) от частичной суммы s отнимаем значение текущего элемента массива: s-mas[i]. 7.2) увеличиваем счётчик i, т.е. увеличиваем номер элемента массива Выход из цикла while произойдет после того, как s станет <0, т.е. разница между начальной суммой sum и накопленной суммой станет <0. 8. Полученное значение переменной i будет являться номером элемента измененного массива, сумма элементов до которого (включая него самого) ближе всего к sum. Выводим i на экран. Так же эффективным будет являтся схожий алгоритм, который ведет накопление частичной суммы и сравнивает ее с начальной. **Алгоритм 2.** mas – массив элементов sum – сумма всех элементов массива s – промежуточная сумма, i – счётчик, k – номер искомого элемента (со значением ближе всего к половине sum, с нижней грани). 1. Считываем число N.

2. Инициализируем переменную sum=0;
3. В цикле for от 1 до N выполняем считывание элементов массива mas[i].
4. Находим сумму элементов массива sum.
5. В цикле for от 1 до N выполняем изменение элементов массива с четными номерами (i mod 2=0) mas[i]=mas[i]*1.4.
Примечание: если позволяет язык программирования, то все эти три действия (2, 3 и 4) выполняем в одном первоначальном цикле (при считывании элементов массива).
5. Присваиваем s=0, k=0.
6. Идем в цикле for от 1 до N пока sum-s>=0 и выполняем в нём следующие действия:
6.1) к частичной сумме s прибавляем значение текущего элемента массива: s=s+mas[i].
6.2) выполняем проверку: если sum-s>=0, то увеличиваем номер элемента массива k;
Если это не так, то выполняем выход из цикла. Т.е.
7. Полученное значение переменной k будет являться номером элемента массива, сумма элементов до которого (включая него самого) ближе всего к sum. Выводим k на экран.

Использование векторов или списков при решении данной задачи является еще более эффективным способом решения задачи.

Другие алгоритмы решения данной задачи (с отниманием и возвратом, с накоплением разниц в других массивах и т.д.) не являются эффективными.

Пример программы на языке C++:

```
#include <iostream>
using namespace std;

int main()
{
    float sum=0,s=0;
    int i, n;
    float mas[10000];
    cin >> n;
    for (i = 0; i < n; ++i) {
          cin >> mas[i];
          sum =sum+mas[i];
          if(i%2==1) mas[i]=mas[i]*1.4;
          cout<<mas[i]<<" ";
     }
     s =sum;
     cout<<s<<endl;
     i = 0;
     while (s >= 0) {
          s=s-mas[i];
          i++;
     }
```

```
    i--;
     cout << i;
     return 0;
}
```

Второй алгоритм решения (к тому же с векторами).

Пример программы на языке C++:

```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    float sum=0,s=0;
    int i, n, k=0;
    cin >> n;
    vector <float>mas(n);
    for (i = 0; i < n; ++i) {
          cin >> mas[i];
          sum +=mas[i];
          if(i%2==1) mas[i]*=1.4;
     }
    s=0;
    for (i = 0; i < n; i++) {
        s+= mas[i];
        if (s<=sum) k++;
        else break;//выход из цикла
    }
    cout << k;
    return 0;
}
```

Пример программы на языке Python:

```
n = int(input())
mas = list(map(float, input().split()))
sm = sum(mas)
s = 0
k = 0
for i in range(len(mas)):
    if (i % 2) ==0 :
         mas[i]*=1.4
    s += mas[i]
    if s <= sm:
        k += 1
    else:
        break
print(k)
```

5.	программа	30	Из условий задачи следует, что программа должна считать массив mas положительных целых чисел размера N (N>2). Нужно перебрать все пары различных элементов последовательности, найти модуль разностей этих пар и определить, сколько среди них являются простыми числами.

Решением задачи является полным перебор всех пар с проверкой модуля разности на простоту. Нет никаких вариантов ускорения алгоритма.
Алгоритм:
mas – массив элементов
s – сумма пары элементов,
i,j,l – счётчики, k – количество модулей разностей пар, являющихся простыми числами
Из условий задачи следует, что программа должна считать массив mas положительных целых чисел размера N (N>2). Нужно перебрать все пары различных элементов последовательности и найти количество таких пар, сумма элементов которых является квадратом некоторого целого числа.
Перебор возможных пар элементов никак не ускорить. А вот проверку на то, что сумма двух элементов является квадратом можно выполнить двумя способами: неэффективным и эффективным.

1 вариант решения. С полным перебором при проверке на квадрат (неэффективный способ).

mas – массив элементов
s – сумма пары элементов,
i,j,d – счётчики, k – количество пар, сумма которых является квадратами
1. Считываем число N.
2. В цикле от 1 до N считываем элементы массива (т.е. числа последовательности).
3. Во вложенных циклах перебираем все возможные пары чисел.
Первый цикл по i перебирает все элементы от 1 до предпоследнего (i – индекс элемента массива, который будет первым в паре). Второй по j перебирает все элементы от следующего за текущим (т.е. j=i+1) до последнего (j – индекс элемента массива, который будут вторым в паре).
3.1) находим модуль разности s=abs(mas[i] - mas[j]).
3.2) Выполняем проверку разности s, что она является простым числом. Для этого в цикле перебираем числа d (от 2 до sqrt(s)), ищем остаток от деления s на d. Если остаток=0, значит d является делителем. Если не найдем ни одного делителя, то число s является простым. В этом случае увеличиваем счётчик k на 1.
4) Повторяем действия для следующих пар.
5) В полученном значении переменной k будет содержаться количество модулей разностей пар, являющимися простыми числами. Выводим значение k на экран.

Использование векторов или списков при решении данной задачи является еще более эффективным способом решения задачи.

Пример программы на языке C++:

```
#include <iostream>
#include <cmath>
using namespace std;

int main() {
    long long n, s=0;
```

```
    int k=0, del=0;

    int mas[10000];
    cin >> n;
    for (int i = 0; i < n; i++) {
        cin >> mas[i];
    }
    for (int i = 0; i < (n - 1); i++) {
        for (int j = i + 1; j < n; j++) {
            s=abs(mas[i]-mas[j]);
             del=0;
            for (int d = 2;d<=pow(s,1.0/2); d++)
            {
               if(s%d==0) del++;
            }
            if (del==0) k++;
        }
    }
    cout << k;
    return 0;
}
```

Пример программы на языке C++ (с вектором):

```
#include <iostream>
#include <cmath>
#include <vector>
using namespace std;

int main() {
    long long n, s=0;
    int k=0, del=0;

    cin >> n;
    vector <long long>mas(n);

    for (int i = 0; i < n; i++) {
        cin >> mas[i];
    }
    for (int i = 0; i < (n - 1); i++) {
        for (int j = i + 1; j < n; j++) {
            s=abs(mas[i]-mas[j]);
             del=0;
            for (int d = 2;d<=pow(s,1.0/2); d++)
            {
               if(s%d==0) del++;
            }
            if (del==0) k++;
        }
    }
    cout << k;
    return 0;
}
```

Пример программы на языке Python:

```
def prostoe(num):
    if num == 1:
        return False
    for i in range(2, int(num**1/2)):
        if num%i == 0:
            return False
    return True


n = int(input())
mas = []
k= 0
for i in range(n):
    mas.append(input())

for i in range(n):
    for j in range(i + 1, n):
        s=abs(int(mas[i])-int(mas[j]))
        if prostoe(s):
          k+=1
print(k)
```

Информатика. 9 класс
Критерии оценивания

Для всех вариантов

Задача 1.

Правильный ответ с полным объяснением – 15 баллов.

Доказательство верное, правильный ответ, пропущен один шаг в доказательстве – 14 баллов.

Правильный ответ, решение подбором с проверкой выполнения – 13 баллов.

Правильный ответ, рассуждение неполное или решение подбором с полным перебором подходящих систем счисления (не оптимальное решение) – 12 баллов.

В целом рассуждение верное, ответ неверный – 11 баллов.

Частично верное рассуждение, ответ верный – 9 баллов.

Частично верное рассуждение, ответ неверный – 7 баллов.

Частично верное рассуждение, ответ неверный, решение идет подбором – 7 баллов.

Ответ подбором, без объяснения или просто проверено выполнение равенства в нужной системе счисления – 5 баллов.

Неверное рассуждение, ответ неверный – 4 баллов.

Правильный ответ без пояснения – 4 балла.

Другой ответ – 0 баллов.

Задача 2.

Правильный ответ с полным объяснением – 20 баллов.

Верное рассуждение, объяснение полное, ответ неверный вследствие арифметической ошибки, допущенной в одном действии – 15 баллов.

Правильный ответ, рассуждение неполное или решение подбором – 12 баллов.

В целом верное рассуждение, ответ неверный вследствие арифметических или логических ошибок – 10 баллов.

Рассуждение неполное, приведены только вычисления, ответ неверный – 9 баллов.

Правильный ответ, рассуждение неверное или приведены только вычисления (причем неверные) – 8 баллов.

Частично верное рассуждение, ответ неверный – 8 баллов.

Неверное рассуждение, неправильный ответ – 5 баллов.

Правильный ответ без пояснения – 3 баллов.

Другой ответ – 0 баллов.

Задача 3.

Правильный ответ и предоставлен файл с электронной таблицей с верными расчётами с использование формул и функций– 10 баллов.

Правильный ответ и предоставлен файл с электронной таблицей с верными расчётами, но недовведенными до финальной стадии – 8 баллов

Неправильный ответ и предоставлен файл с электронной таблицей с верными расчётами – 8 баллов

Правильный ответ и предоставлен файл с электронной таблицей с верными расчётами, но без использования формул и функций – 7 баллов

Правильный ответ и предоставлен файл с электронной таблицей с неверными расчётами – 6 баллов.

Неправильный ответ и предоставлен файл с электронной таблицей с неверными расчётами – 6 баллов.

Неправильный ответ и предоставлен файл с электронной таблицей с частично верными расчётами (не учтено одно из ограничений задачи) – 6 баллов.

Неправильный ответ и предоставлен файл с электронной таблицей с частично верными расчётами – 5 баллов.

Неправильный ответ и предоставлен файл с электронной таблицей с неверными расчётами – 4 балла.

Правильный ответ и не предоставлен файл с электронной таблицей, есть программа с верными расчётами – 4 балла.

Неправильный ответ и не предоставлен файл с электронной таблицей, есть программа с неверными расчётами – 3 балла.

Неправильный ответ и предоставлен файл с электронной таблицей, без расчетов – 2 балла.

Неправильный ответ и не предоставлен файл с электронной таблицей, есть вычисления вручную на бланке, но неверные – 2 балла.

Правильный ответ и не предоставлен файл с электронной таблицей с расчётами – 2 балла.

Ответ близок к правильному и не предоставлен файл с электронной таблицей с расчётами – 1 балл.

Другой ответ – 0 баллов.

Задача 4.

Правильно решающий задачу, работающий и эффективный программный код – 25 баллов.

Правильно решающий задачу, работающий и неэффективный программный код – 23 балла.

Работающий и эффективный программный код, но есть незначительные ошибки в работе алгоритма – 18 баллов

Программный код работает, но он неэффективный и есть незначительные ошибки в работе алгоритма (частично верный код) – 16 баллов

Программный код частично верный, но не работающий – 12 баллов

Программный код работающий, но большая часть алгоритма ошибочна – 10 баллов

Программный код работающий, но полностью ошибочный – 8 баллов

Программный код неверный и не работающий – 5 баллов

Описан алгоритм работы программы, но не написан программный код – 2 балла.

Другой ответ – 0 баллов.

Задача 5.

Правильно решающий задачу, работающий и эффективный программный код – 30 баллов.

Правильно решающий задачу, работающий и неэффективный программный код – 28 баллов.

Работающий и эффективный программный код, есть незначительные ошибки в работе алгоритма – 25 баллов.

Работающий и неэффективный программный код, есть незначительные ошибки в работе алгоритма – 22 балла

Программный код работает, частично верный код, есть значительные ошибки в работе алгоритма – 15 баллов

Программный код частично верный, но не работающий – 13 баллов

Программный код работающий, но полностью ошибочный – 10 баллов

Программный код работающий, но решающий другую задачу (включая другой вариант) – 9 баллов

Программный код неверный и не работающий – 6 баллов

Описан алгоритм работы программы, но не написан программный код – 3 балла.

Другой ответ – 0 баллов

Информатика. 10 класс
Решения

<table>
<tr><th colspan="4">1 вариант</th></tr>
<tr><th>№</th><th>Ответ</th><th>Балл</th><th>Решение</th></tr>
<tr><td>1</td><td>23</td><td>20</td><td>Рассмотрим остатки при делении числа 49 на 13 и 17, это 10 и 15. Неизвестное нам число x на предыдущем шаге представлялось как x=y*13+z и x=p*17+q.
Если z>q, то 49= x+z= y*13+z+z= y*13+2*z.
Если z<q, то 49= x+q= y*17+q+q= y*17+2*q.
Другими словами, один из этих остатков удвоили.
Остаток 10 по модулю 13 всё равно, что 23, а 15 по модулю 17 всё равно, что 32.
Так как 23 является нечетным числом, то значит, первый остаток z (по модулю 13) не удваивался.
Второй остаток равен 32, значит, именно этот остаток удваивался. Получается, что начальный остаток q=16.
Так как 49= x+q=x+16, то x=33.
К полученному числу 33 нужно ещё раз применить этот же трюк: неизвестное нам число x на предыдущем шаге представлялось как x=y*13+z и x=p*17+q.
Если z>q, то 33= x+z= y*13+z+z= y*13+2*z.
Если z<q, то 33= x+q= y*17+q+q= y*17+2*q.
Остатки при делении числа 33 на 13 и 17, это 7 и 16.
Остаток 7 по модулю 13 всё равно, что 20, а 16 по модулю 17 всё равно, что 33.
Так как 33 является нечетным числом, то значит, второй остаток (по модулю 17) не удваивался.
Первый остаток равен 20, значит, именно этот остаток удваивался.
Получается, что начальный остаток z=10.
Так как 33= x+z=x+10, значит x=23.
Итак, в начале было одно число: 23.
Ответ: 23</td></tr>
<tr><td>2</td><td>Первый игрок, Бельчонок</td><td>15</td><td>По условиям игры, выигрывает игрок, после хода которого сумма координат становится не менее 46. Выигрышная стратегия есть у Бельчонка. Он может первым ходом утроить координату x и привести фигуру на клетку (12,2). Отсюда Ёжику нельзя двигать фигуру по первой координате, так как если он сдвинет на один или, тем более, утроит координату – у бельчонка либо следующий ход будет победным, либо победа будет через 1 ход. Поэтому логично начать передвигать по второй координате. Но и в этих ситуациях Бельчонок может воспользоваться преимуществом первого хода (предвыигышной ситуации для Ёжика). То есть победит Бельчонок. Продемонстрируем выигрышную стратегию Бельчонка при любых ответных ходах противника. Обозначим Б – Бельчонок, Е – Ёжик
Обозначим ПБ – победа Бельчонка, ПЕ – победа Ёжика. Красным цветом будем помечать проигрышные для Бельчонка ситуации

<table>
<tr><td>Ход Б</td><td>Ход Е</td><td>Ход Б</td><td>Ход Е</td><td>Ход Б</td><td>Ход Е</td><td>Ход Б</td></tr>
<tr><td>12,2</td><td>36,2</td><td>108, 2 ПБ</td><td></td><td></td><td></td><td></td></tr>
</table>
</td></tr>
</table>

				13,6	39,6	117,6 ПБ		
					13,18	13,54 ПБ		
					13,7	39,7 ПБ		
					14,6	42,6 ПБ		
			12,6	12,7	36,7	108,7 ПБ		
					12,21	12,63 ПБ		
					13,7	39,7 ПБ		
					12,8	12,9	12,10	16,10 ПБ
				36,6	ПЕ			
				12,18	ПЕ			
			13,2	13,6	14,6	42,6 ПБ		
					13,7	39,7 ПБ		
					39,6	117,6 ПБ		
					13,18	13,54 ПБ		
				В остальных – ПЕ				
			12,3	12,9	13,9	39,7 ПБ		
					12,10	36,10 ПБ		
					39,6	117,6 ПБ		
					12,27	12,81 ПБ		
				В остальных – ПЕ				

В ветке с первым ходом Бельчонка (4,6) также можно построить выигрышную

			стратегию, а вот в остальных случаях существуют целые ветки с выигрышами Ёжика. Ответ: Победит Бельчонок (первый игрок), его первый ход (4,2)->(12,2). Выигрышная стратегия указана в таблице.
3	414	10	Из условия задачи следует, что нам нужно найти максимальное количество шишек, которое может собрать Бельчонок при заданных ограничениях, пройдя из **третьей слева верхней клетки** в **правую нижнюю**. Движение начинается из ячейки C1, поэтому столбцами A и B можно пренебречь. В условиях задачи имеются два ограничения: 1) при каждом сборе урожая (т.е. фактически при заходе в клетку) Бельчонок тратит энергию, которую он восстанавливает, съедая одну кедровую шишку. 2) собрать из каждой ячейки и унести с собой можно не более 18 шишек, т.е. если в ячейке находится число больше 18, то взять из нее мы можем только 18. Обратите внимание, что мы будем искать максимальные значения и заходить в каждую клетку. Т.е. нужно первым действием отнять 1 из каждой ячейки. После этого вторым действием нужно проставить в каждую клетку со значением >18, значение 18. Нельзя было менять эти действия местами, так как неправильно заменятся значения. (Например, было 25. 25-1=24. Потом заменим его на 18, поскольку мы можем из 24 забрать только 18. Если сделать эти действия в обратном порядке, то 25 заменятся на 18, потом 18-1=17. Но это неверно! Ведь в ячейке было >18. Значит, мы могли 18 забрать из этой ячейки). Оптимальнее всего объединить эти два действия в одно. Таким образом, для решения этой задачи можно пользоваться двумя способами. **Первый способ (эффективный):** 1) Исходные данные находятся в области C1:N12. Из исходной таблицы получим измененную с помощью формулы =ЕСЛИ(C1<19;C1-1;18) – это делаем для каждой ячейки. Здесь мы сразу отнимаем 1, и все значения > 18 заменяем на 18. Пусть новые значения нашего диапазона находятся в области C14:N25. 2) Теперь будем искать максимальное значение. Для поиска максимального значения будем работать с областью C27:N38, при расчетах будем использовать значения количества шишек в области C14:N25. В ячейку C27 напишем значение **=C14**. Для каждой ячейки левого столбца это будет сумма всех ячеек выше текущей. Внесем в ячейку C28 формулу **=C15+C27** и скопируем за маркер вниз до ячейки C38. Для каждой ячейки строки №27 это будет сумма всех ячеек левее текущей. Внесем в ячейку B27 формулу **=D14+C27** и скопируем за маркер вправо до ячейки N27. Далее в ячейку D28 вставим формулу **= =D15+МАКС(C28;D27).** Т.е. мы суммируем значение, которое находится в данной ячейке, и максимальное число, полученное на предыдущем шаге (придя из соседних по движению, так как в каждую ячейку можно прийти либо слева, либо сверху). **С**копируем эту формулу, потянув за маркер в ячейки D28:N38. Значение в ячейке N38 будет максимальным количеством шишек, которое может собрать Бельчонок — 414. **Второй способ (менее эффективный):** В этом алгоритме пункт 1) предыдущего алгоритма разбивается на два шага: Из исходной таблицы получим измененную с помощью формулы =A1-1 – это делаем для каждой ячейки. В измененной таблице все значения уменьшатся на

			1. Например, новые значения нашего диапазона находятся в строках с A14 по N25. 2) Затем из измененной таблицы получаем вторую измененную с помощью формулы =ЕСЛИ(A14>18;18;A14) – это делаем для каждой ячейки. В измененной таблице все значения>18 заменятся на 18. Все остальные действия алгоритма схожи с первым алгоритмом (только со сдвигом по строкам из-за второй промежуточной таблицы). Ответ: 414
4		25	Нужно найти в массиве (*или векторе, списке*) два таких числа, удаление которых из массива минимально изменяет среднее значение новой полученной последовательности. Если существует несколько таких пар, то требуется вывести одну любую такую пару. На первый взгляд, кажется, что эта задача решается удалением из массива двух чисел наиболее близко лежащими к среднему значению, или, наоборот, двух наиболее отдаленных элементов. Но, к сожалению, это не так. Может оказаться, что наиболее сильно влияет пара элементов, находящихся в произвольных местах массива. Приведем пример такого массива: 1 2 3 30 70 100 Среднее=34.33 Если удалить 1 и 2, то среднее=50.75 Если удалить 3 и 30, то среднее=43.25 Если удалить 70 и 100, то среднее=9 Если удалить 1 и 70, то среднее=33.75. Оно ближе всего к 34.33. Данную задачу можно решить несколькими способами. Рассмотрим самый оптимальный из них. **Способ решения задачи**: mas – массив элементов sum – сумма всех элементов массива x, y – пара элементов, наименее изменяющие среднее sred – текущее среднее значение (без пары x, y) minsred – минимальное среднее, i, j – счётчики. 1. Считываем число N. 2. Инициализируем переменную sum=0; 3. В цикле for от 1 до N выполняем считывание элементов массива mas[i]. (Если позволяет язык программирования, то сразу же накапливаем сумму элементов массива: sum=sum+mas[i].) 4. Если суммирование не выполнялось при считывании, то выполняем его сейчас. 5. Присваиваем x=mas[0], y=mas[1], minsred=(sum-x-y)/(n-2). 6. Во вложенных циклах перебираем все возможные пары чисел. Первый цикл по i перебирает все элементы от 1 до предпоследнего (i – индекс элемента массива, который будет первым в паре). Второй по j перебирает все элементы от следующего за текущим (т.е. j=i+1) до последнего (j – индекс элемента массива, который будут вторым в паре).

			3.1) находим сумму текущее среднее по формуле sred =(sum-mas[i] + mas[j])/(n-2).
			3.2) Выполняем проверку среднего sred, что оно дает наименьшее отклонение от первоначального среднего, чем хранимое минимальное (в переменной minsred). Т.е. сравниваем abs(sum/n- sred)<abs(sum/n-minsred). Если так, то запоминаем x=mas[i], y=mas[j], minsred=sred.
			4) Повторяем действия для следующих пар.
			5) В полученных значениях переменных x y будет содержаться пара, удаление которых из массива минимально изменяет среднее значение новой последовательности. Выводим значения x и y на экран.
			
			Использование векторов или списков при решении данной задачи является еще более эффективным способом решения задачи.
			Допускаются и другие варианты решения задачи, приводящие к тем же результатам.
			Другие алгоритмы решения данной задачи (с отниманием и возвратом элементов, с накоплением разниц в других массивах и т.д.) не являются эффективными.
			
			Пример программы на языке C++:
			#include <iostream>
			using namespace std;
			
			int main()
			{
			long long n, x, y, sum,sred,minsred;
			int i,j;
			cin >> n;
			int mas[10000];
			for (i = 0; i < n; i++) {
			cin >> mas[i];
			sum += mas[i];
			}
			x=mas[0];
			y=mas[1];
			minsred=(sum-x-y)/(n-2);
			for (i = 0; i < n-1; i++)
			for (j = i+1; j < n; j++) {
			sred=(sum-mas[i]-mas[j])/(n-2);
			if (abs(sum/n- sred)<abs(sum/n-minsred))
			{
			minsred=sred;
			x=mas[i];
			y=mas[j];
			}
			}

```
    cout << x<<" "<<y;
    return 0;
}
```

Пример программы на языке C++ (с векторами):

```
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    long long n, x, y, sum,sred,minsred;
    int i,j;
    cin >> n;
    vector <long long>mas(n);
    for (i = 0; i < n; i++) {
        cin >> mas[i];
        sum += mas[i];
    }
    x=mas[0];
    y=mas[1];
    minsred=(sum-x-y)/(n-2);
    for (i = 0; i < n-1; i++)
        for (j = i+1; j < n; j++) {
            sred=(sum-mas[i]-mas[j])/(n-2);
            if (abs(sum/n- sred)<abs(sum/n-minsred))
            {
                minsred=sred;
                x=mas[i];
                y=mas[j];
            }
    }
    cout << x<<" "<<y;
    return 0;
}
```

Пример программы на языке Python:

```
n=int(input())
mas=list(map(int, input().split()))
summ=sum(mas)
x=mas[0]
y=mas[1]
minsred=(summ-x-y)/(n-2)
for i in range(n-1):
    for j in range(i+1,n):
        sred=(summ-mas[i]-mas[j])/(n-2)
        if abs(summ/n- sred)<abs(summ/n-minsred):
```

			minsred=sred x=mas[i] y=mas[j] print(x, y)
5	Написанная программа	30	Имеется матрица из N строк и M столбцов. В каждом элементе матрицы может находиться одно из двух значений: 0 или 1. Нужно найти размер самого большого «острова» из соединенных вместе единичек (находящихся в соседних клетках). Данную задачу можно решить эффективным и и неэффективным способом: 1) рекурсивный алгоритм – эффективный алгоритм. 2) переборный алгоритм с метками – менее эффективный алгоритм. **Алгоритм с рекурсией (эффективный):** Введем дополнительный массив логического типа **c**, в котором будем отмечать уже просмотрен- ные элементы. 2. Идем по матрице размера nXm - во вложенных циклах перебираем номера строк и столбцов. Первый цикл по I перебирает номера строк. Второй по j перебирает номера столбцов. Для каждого элемента a[i,j]проверяем: если элемент еще не посещался (c[i,j]==false) и в нём находится 1, то вычисляется максимум среди текущего максимума и вызова рекурсивной функции rek, которая вычисляет размер «острова» из единичек, начиная с текущих индексов i,j. Рекурсивная функция rek работает так: Для присланной точки в матрице с индексами x, y она проверяет, что индексы его соседей не выходят за границы матрицы (>0, <n, >0, <m) и проверяет соседей данного элемента. И если сосед =1, то вызывает эту же функцию rek рекурсивно для соседа и увеличивает размер «острова». Функция возвращает найденный размер «острова» из единичек. **Алгоритм переборный, без рекурсии (неэффективный):** 1. Объявляем переменную-метку label. Присваиваем ей значение 2. 2. Объявляем переменную логического типа flag – для отслеживания увеличения размера текущего «острова». 3. Идем в цикле do-while или repeat-until, пока флаг=true 4. Внутри цикла flag=false 5. Перебираем строки и столбцы, находим первую 1, ставим на ее место label и flag=true. 6. Объявляем переменную логического типа flag_label. Создаем цикл while (flag_label), в котором еще два цикла for, в которых перебираем строки и столбцы. Для каждого элемента a[i,j] проверяем: если a[i,j]=1 и один из его соседей=label, то a[i,j]=label, flag_label=true. 7. Если внутри циклов for мы все перебрали, остров закончился – изменений не

было, то flag_label будет false, тогда мы выйдем из цикла while и увеличивается значение label на 1.
8. Когда после прохождения циклов flag остался false, то все острова найдены и цикл останавливается.
9. В конце нужно пройти с помощью вложенных циклов for по всем элементам массива. Посчитать количество 2,3,4 и т.д. Максимальное значение как раз покажет нам величину самого большого связного «острова».

Приведем пример реализации эффективного алгоритма.

Пример программы на языке C++ (с векторами):

```
#include<bits/stdc++.h>
using namespace std;

long long rek(long long x, long long y, long long n, long long m,
vector<vector<long long>> &a, vector<vector<bool>> &c) {

    c[x][y] = true;
    long long v = 1;
    if (x>0)
        if (a[x - 1][y] == 1 and !c[x - 1][y]) {
            v += rek(x - 1, y, n, m, a, c);
        }
    if (y>0)
        if (a[x][y - 1] == 1 and !c[x][y - 1]) {
            v += rek(x, y - 1, n, m, a, c);
        }
    if (x<n-1)
        if (a[x + 1][y] == 1 and !c[x + 1][y]) {
            v += rek(x + 1, y, n, m, a, c);
        }
    if (y<m-1)
        if (a[x][y + 1] == 1 and !c[x][y + 1]) {
            v += rek(x, y + 1, n, m, a, c);
        }
    return v;
}
int main() {
    long long n, m;
    cin >> n >> m;
    vector<vector<long long>> a;
    vector<vector<bool>> c;
    a.resize(n, vector<long long>(m , 0));
    c.resize(n, vector<bool>(m, false));
    for (long long i = 0; i < n; i++) {
        for (long long j = 0; j < m; j++) {
            cin >> a[i][j];
        }
    }
    long long maxs = 0;
    for (long long i = 0; i < n; i++) {
        for (long long j = 0; j < m; j++) {
```

```
            if (!c[i][j] and a[i][j] == 1) {
                maxs = max(maxs, rek(i, j, n, m, a, c));
            }
        }
    }
    cout << maxs;
    a.clear();
    c.clear();
    return 0;
}
```

Пример программы на языке Python:

```
n, m = map(int, input().split())
g = [[int(j) for j in input().split()] for i in range(n)]
visit = [[0 for j in range(m)] for i in range(n)]

def rek(i, j) :
    ans = 1
    if g[i][j]==0:
        return 0
    visit[i][j] = 1
    if i>0:
        if visit[i-1][j]==0: ans += rek(i-1, j)
    if i<n-1:
        if visit[i+1][j]==0: ans += rek(i+1, j)
    if j>0:
        if visit[i][j-1]==0: ans += rek(i, j-1)
    if j<m-1:
        if visit[i][j+1]==0: ans += rek(i, j+1)
    return ans

mx = 0
for i in range(n):
    for j in range(m):
        if g[i][j]==1 and visit[i][j]==0:
            mx = max(rek(i, j), mx)

print(mx)
```

Допускаются и другие варианты решения задачи, приводящие к тем же результатам.

2 Вариант

№	Ответ	Балл	Решение

1	27 и 28	20	Рассмотрим остатки при делении числа 42 на 11 и 13, это 9 и 3. Неизвестное нам число x на предыдущем шаге представлялось как x=y*11+z и x=p*13+q. Если z>q, то 42= x+z= y*11+z+z= y*11+2*z. Если z<q, то 42= x+q= y*13+q+q= y*13+2*q. Другими словами, один из этих остатков удвоили. Остаток 9 по модулю 11 всё равно, что 20, а 3 по модулю 13 всё равно, что 16. 1) Так как 20 является четным числом, то возможно удваивался первый остаток z (по модулю 11). Если это так, то получается, что начальный остаток z=10. Так как 42= x+z=x+10, то x=32. К полученному числу 32 нужно ещё раз применить этот же трюк: неизвестное нам число x на первом шаге также представлялось как x=y*11+z и x=p*13+q. Если z>q, то 32= x+z= y*11+2*z. Если z<q, то 32= x+q= y*13+2*q. Остатки при делении числа 32 на 11 и 13, это 10 и 6. 1.1) Первый остаток равен 10, возможно этот остаток удваивался. Получается, что начальный остаток z=5. Так как 32= x+z=x+5, значит x=27. 1.2) Второй остаток 6, возможно этот остаток удваивался. Получается, что начальный остаток q=3. Так как 32= x+q=x+3, значит x=29. Но для 29 не выполняется условие, что z<q, так как 29=2*11+7 и 29=2*13+3. А остаток 6 по модулю 13 всё равно, что 19 - нечётное число, не подходит. Таким образом, мы определили, что первоначальным числом в первом случае могло быть 27 2) Второй остаток – при делении числа 42 на 13 – был равен 16. Если этот остаток удваивался, то получается, что начальный остаток q=8. Так как 42= x+q=x+8, то x=34. К полученному числу 34 снова применяем действия по нахождению остатков: если z>q, то 34= x+z=y*11+2*z. Если z<q, то 33= x+q= y*13+2*q. Остатки при делении числа 34 на 11 и 13, это 1 и 8. Остаток 1 по модулю 11 всё равно, что 12. 2.1) Первый остаток равен 12, значит, именно этот остаток удваивался. Получается, что начальный остаток z=6. Так как 34= x+z=x+6, значит x=28. 2.2) Второй остаток 8, возможно этот остаток удваивался. Получается, что начальный остаток q=4. Так как 34= x+q=x+4, значит x=30. Но для 30 не выполняется условие, что z<q, так как 30=2*11+8 и 30=2*13+4. А остаток 4 по модулю 13 всё равно, что 17 - нечётное число, не подходит. Таким образом, мы определили, что первоначальным числом во втором случае могло быть 28. Ответ: 27 и 28.
2	Первый игрок	15	По условиям игры, победителем считается игрок, после хода которого, сумма орехов в обеих кучах будет 63 или больше. Выигрышная стратегия есть у первого игрока. Он может первым ходом утроить координату x и сделать значение в кучах (18,4). В этой ситуации второму игроку нельзя двигать увеличивать первую кучу, так как если он сдвинет на один или, тем более, утроит – у первого игрока либо следующий ход будет победным, либо победа будет через 1 ход. Поэтому логично начать увеличивать вторую кучу. Но и в этих ситуациях первый игрок может воспользоваться преимуществом первого

хода (т.е. добавлять так, что утроение наибольший кучи не давало победы 2 игроку). То есть победит первый игрок. Продемонстрируем выигрышную стратегию первого игрока при любых ответных ходах противника. Обозначим И1 – Бельчонок, И2 – Ёжик

Обозначим П1 – победа игрока 1, П2 – победа игрока 2. Красным цветом будем помечать проигрышные для игрока 1 ситуации.

<table>
<tr><th>Ход И1</th><th>Ход И2</th><th>Ход И1</th><th>Ход И2</th><th>Ход И1</th><th>Ход И2</th><th>Ход И1</th></tr>
<tr><td rowspan="20">18,4</td><td>54,4</td><td>162,4
П1</td><td></td><td></td><td></td><td></td></tr>
<tr><td rowspan="2">18,12</td><td>54,12
П1</td><td></td><td></td><td></td><td></td></tr>
<tr><td colspan="2">В остальных – П2</td><td></td><td></td><td></td></tr>
<tr><td rowspan="5">19,4</td><td rowspan="4">19,5</td><td>19,6</td><td>57,6
П1</td><td></td><td></td></tr>
<tr><td>20,5</td><td>60,5
П1</td><td></td><td></td></tr>
<tr><td>57,5</td><td>171,5
П1</td><td></td><td></td></tr>
<tr><td>19,15</td><td>57,6
П1</td><td></td><td></td></tr>
<tr><td colspan="2">В остальных – П2</td><td></td><td></td><td></td></tr>
<tr><td rowspan="14">18,5</td><td rowspan="13">18,6</td><td>18,7</td><td>18,8</td><td>18,9</td><td>54,9
П1</td></tr>
<tr><td></td><td></td><td>19,8</td><td>57,8
П1</td></tr>
<tr><td></td><td></td><td>54,8</td><td>162,8
П1</td></tr>
<tr><td></td><td></td><td>18,24</td><td>18,72
П1</td></tr>
<tr><td></td><td colspan="2">В остальных – П2</td><td></td></tr>
<tr><td>19,6</td><td>57,6
П1</td><td></td><td></td></tr>
<tr><td>54,6</td><td>162,6
П1</td><td></td><td></td></tr>
<tr><td>18,18</td><td>54,18
П1</td><td></td><td></td></tr>
<tr><td colspan="2">В остальных – П2</td><td></td><td></td><td></td></tr>
</table>

В ситуациях с возможными другими значениями первого хода первого игрока существуют целые ветки с выигрышами второго игрока.

Ответ: Победит первый игрок, его первый ход (6,4)->(18,4). Выигрышная

			стратегия указана в таблице.
3	378	10	Из условия задачи следует, что нам нужно найти максимальное количество моркови, которое может собрать Зайчик при заданных ограничениях, пройдя из **правой нижней клетки** в **левую верхнюю**. Причём движение начинается из ячейки N14, поэтому столбцом О можно пренебречь. В условиях задачи имеются два ограничения: 1) кроты похитили ровно по одной морковке из каждой клетки, в которой первоначально было нечётное значение. 2) с каждой клетки Зайчик может собрать и унести с собой не более 14 морковок, т.е. если в ячейке находится число больше 14, то взять из нее мы можем только 14. Таким образом, перед началом движения нужно первым действием отнять 1 из ячеек, в которых содержатся нечётные значения. После этого вторым действием нужно проставить в каждую клетку со значением >14, значение 14. Оптимальнее всего объединить эти два действия в одно. Таким образом, для решения этой задачи можно пользоваться двумя способами. Первый способ (менее эффективный): 1) Исходные данные находятся в области A1:N14. Из исходной таблицы получим измененную с помощью формулы =ЕСЛИ(ОСТАТ(A1;2)=1;A1-1;A1) – это делаем для каждой ячейки. Здесь мы сразу отнимаем 1 у всех нечётных чисел (с остатком 1 при делении на 2). Пусть измененные значения находятся в ячейках A16:N29. 2) Из предыдущей таблицы получим еще одну измененную с помощью формулы =ЕСЛИ(A16<14;A16;14) – это делаем для каждой ячейки. Здесь мы все значения > 14 заменяем на 14. Пусть полученные измененные значения находятся в ячейках A31:N44. 2) Теперь будем искать максимальное значение. Для поиска максимального значения будем работать с областью A46:N59, при расчетах будем использовать значения в вышеполученной области A31:N44. В ячейку N59 напишем значение **=N44**. Для каждой ячейки правого столбца это будет сумма всех ячеек ниже текущей. Внесем в ячейку N58 формулу **=N43+N59** и скопируем за маркер вверх до ячейки N46. Для каждой ячейки строки №59 это будет сумма всех ячеек левее текущей. Внесем в ячейку M59 формулу **=M44+N59** и скопируем за маркер вправо до ячейки A59. Далее в ячейку M58 вставим формулу **= =M43+МАКС(M59;N58).** Т.е. мы суммируем значение, которое находится в данной ячейке, и максимальное число, полученное на предыдущем шаге (придя из соседних по движению, так как в каждую ячейку можно прийти либо справа, либо снизу). **С**копируем эту формулу, потянув за маркер в ячейки A46:M58. Значение в ячейке A46 будет максимальным количеством морковок, которое может собрать Зайчик — 378. **Второй способ (эффективный):** В этом алгоритме объединяются пункты 1) и 2) предыдущего алгоритма. Вместо двух промежуточных таблиц получаем только одну с помощью формулы: =ЕСЛИ(A1>14;14;ЕСЛИ(ОСТАТ(A1;2)=1;A1-1;A1)) – это делаем для каждой ячейки. Здесь мы сразу все значения > 14 заменяем на 14 и отнимаем

			1 от нечётных чисел<14. Потому что при отнимании 1 у нечетных чисел>14 (15,17,...) получаемые числа (14,16,..) всё равно >=14, а значит, будут просто заменены 14. Поэтому для любых чисел>14 (четных и нечётных) можно просто выполнить только одно действие – замену на 14. Все остальные действия алгоритма по поиску максимального значения схожи с первым алгоритмом. Ответ: 378
4	Написанная программа	25	Нужно найти в массиве (*или векторе, списке*) два таких числа, удаление которых из массива максимально изменяет среднее значение новой полученной последовательности. Если существует несколько таких пар, то требуется вывести одну любую такую пару. Данную задачу можно решить двумя способами: 1) переборным – перебирая все пары элементы, находя новое среднее, вычисляя отклонение от начального среднего и запоминая значение, дающее максимальное отклонение – неэффективный алгоритм. 2) найти два минимума и два максимума, вычислить два новых средних (без минимумов и максимумов), и выбрать из минимумов или максимумов, то, что даст максимальное отклонение от начального среднего – эффективный алгоритм. **Первый алгоритм (переборный, неэффективный)**: mas – массив элементов sum – сумма всех элементов массива x, y – пара элементов, наиболее изменяющие среднее sred – текущее среднее значение (без пары x, y) maxsred – максильное среднее, i, j – счётчики. 1. Считываем число N. 2. Инициализируем переменную sum=0; 3. В цикле for от 1 до N выполняем считывание элементов массива mas[i]. (Если позволяет язык программирования, то сразу же накапливаем сумму элементов массива: sum=sum+mas[i].) 4. Если суммирование не выполнялось при считывании, то выполняем его сейчас. 5. Присваиваем x=mas[0], y=mas[1], maxsred=(sum-x-y)/(n-2). 6. Во вложенных циклах перебираем все возможные пары чисел. Первый цикл по i перебирает все элементы от 1 до предпоследнего (i – индекс элемента массива, который будет первым в паре). Второй по j перебирает все элементы от следующего за текущим (т.е. j=i+1) до последнего (j – индекс элемента массива, который будут вторым в паре). 3.1) находим сумму текущее среднее по формуле sred =(sum-mas[i] + mas[j])/(n-2). 3.2) Выполняем проверку среднего sred, что оно дает наибольшее отклонение от первоначального среднего, чем хранимое максимальное (в переменной maxsred). Т.е. сравниваем abs(sum/n- sred)>abs(sum/n-maxsred). Если так, то

			запоминаем x=mas[i], y=mas[j], maxsred=sred. 4) Повторяем действия для следующих пар. 5) В полученных значениях переменных x y будет содержаться пара, удаление которых из массива максимально изменяет среднее значение новой последовательности. Выведем на экран значения x и y. **Второй алгоритм (оптимальный):** mas – массив элементов sum – сумма всех элементов массива max1, max2 – два максимальных элемента, min1, min2 – два минимальных элемента srMIN – среднее без минимумов, srMAX – среднее без минимумов, i – счётчик. 1. Считываем число N. 2. Инициализируем переменную sum=0; max1 = 0, max2 = 0, min1 = 100000000, min2 = 100000000; 3. В цикле for от 1 до N выполняем считывание элементов массива mas[i]. (Если позволяет язык программирования, то сразу же накапливаем сумму элементов массива: sum=sum+mas[i] и ищем минимумы и максимумы. Если язык не позволяет это сделать, значит после считывания будет еще один цикл, в котором выполняется суммирование и поиск двух максимумов и двух минимумов. 3.1) поиск максимума: для текущего элемента mas[i] выполняем сравнение если max1 < mas[i], то max2 = max1 и max1 = mas[i]. Если это не так, то наше число меньше первого максимума, но возможно оно является вторым по значению максимумом? Проверяем это: если max2 < mas[i], то max2 = mas[i]. 3.2) Схожим образом выполняется поиск двух минимумов: для текущего элемента mas[i] выполняем сравнение если min1 > mas[i], то min2 = min1 и min1 = mas[i]. Если это не так, и элемент mas[i] больше первого минимума, но возможно он является вторым по значению минимумом? Проверяем это: если(min2 > mas[i]), то min2 = mas[i]. 4. После выхода из цикла мы знаем sum, max1,max2,min1,min2. 5. Найдем среднее без минимумов srMIN = (sum - min1 - min2) / (n - 2); 6. Найдем среднее без максимумов srMAX = (sum - max1 - max2) / (n - 2); 7. Выберем из srMIN и srMAX то, что даст максимальное отклонение от начального среднего если (abs(sum/n - srMIN) > abs(sum/n - srMAX)), то выведем min1 и min2, иначе - max2 и max1. Использование векторов или списков при решении данной задачи является еще более эффективным способом решения задачи. Допускаются и другие варианты решения задачи, приводящие к тем же результатам.

Другие алгоритмы решения данной задачи (с отниманием и возвратом элементов, с накоплением разниц в других массивах и т.д.) не являются эффективными.

Пример программы на языке C++:

```
#include <iostream>
using namespace std;

int main() {
   int n, max1 = 0, max2 = 0, min1 = 100000000, min2 = 100000000;
   double sum = 0;
   cin >> n;
   int mas[10000];
   for (int i = 0; i < n; ++i) {
      cin >> mas[i];
      sum += mas[i];
      if (max1 < mas[i]) {
         max2 = max1;
         max1 = mas[i];
      } else if (max2 < mas[i]) {
         max2 = mas[i];
      }
      if (min1 > mas[i]) {
         min2 = min1;
         min1 = mas[i];
      } else if (min2 > mas[i]) {
         min2 = mas[i];
      }
   }

   double srMIN, srMAX;
   if (n == 2) {
      cout << mas[0] << ' ' << mas[1];
   } else {
      srMIN = (sum - min1 - min2) / (n - 2);
      srMAX = (sum - max1 - max2) / (n - 2);
      if (abs(sum/n - srMIN) > abs(sum/n - srMAX)) {
         cout << min1 << ' ' << min2;
      } else {
         cout << max2 << ' ' << max1;
      }
   }
   return 0;
}
```

Пример программы на языке C++ (с векторами):

```
#include <iostream>
```

```
#include <vector>
using namespace std;
using namespace std;

int main() {
   int n, max1 = 0, max2 = 0, min1 = 100000000, min2 = 100000000;
   double sum = 0;
   cin >> n;
   vector<int> mas(n);
   for (int i = 0; i < n; ++i) {
      cin >> mas[i];
      sum += mas[i];
      if (max1 < mas[i]) {
         max2 = max1;
         max1 = mas[i];
      } else if (max2 < mas[i]) {
         max2 = mas[i];
      }
      if (min1 > mas[i]) {
         min2 = min1;
         min1 = mas[i];
      } else if (min2 > mas[i]) {
         min2 = mas[i];
      }
   }



   double srMIN, srMAX;
   if (n == 2) {
      cout << mas[0] << ' ' << mas[1];
   } else {
      srMIN = (sum - min1 - min2) / (n - 2);
      srMAX = (sum - max1 - max2) / (n - 2);
      if (abs(sum/n - srMIN) > abs(sum/n - srMAX)) {
         cout << min1 << ' ' << min2;
      } else {
         cout << max2 << ' ' << max1;
      }
   }
   return 0;
}
```

Пример программы на языке Python:

```
n=int(input())
mas=list(map(int, input().split()))
summ=sum(mas)

max1 = 0
```

			max2 = 0
			min1 = 100000000
			min2 = 100000000;
			
			for i in range(n):
			  if max1 < mas[i]:
			  max2 = max1
			  max1 = mas[i]
			  elif max2 < mas[i]:
			  max2 = mas[i]
			
			  if min1 > mas[i]:
			  min2 = min1
			  min1 = mas[i]
			  elif min2 > mas[i]:
			  min2 = mas[i]
			
			srMIN = (summ - min1 - min2) / (n - 2)
			srMAX = (summ - max1 - max2) / (n - 2)
			if abs(summ/n - srMIN) > abs(summ/n - srMAX):
			  print(min1,min2)
			else:
			  print(max2,max1)

5	Написанная программа	30	Имеется матрица из N строк и M столбцов. В каждом элементе матрицы может находиться одно из двух значений: 0 или 1. Нужно найти размер самого большого «острова» из соединенных вместе единичек (находящихся в соседних клетках). Данную задачу можно решить эффективным и и неэффективным способом: 1) рекурсивный алгоритм – эффективный алгоритм. 2) переборный алгоритм с метками – менее эффективный алгоритм. **Алгоритм с рекурсией (эффективный):** 1. Введем дополнительный массив логического типа c, в котором будем отмеч уже просмотренные элементы. 2. Идем по матрице размера nXm - во вложенных циклах перебираем номера строк и столбцов. Первый цикл по I перебирает номера строк. Второй по j перебирает номера столбцов. Для каждого элемента a[i,j]проверяем: если элемент еще не посещался (c[i,j]=false) и в нём находится 1, то вызываем рекурсивную функцию rek, которая помечает остров, начиная с текущих индексов i,j. Рекурсивная функция rek работает так: Для присланной точки в матрице с индексами x, y она проверяет, что эти индексы не выходят за границы матрицы (x>0, x<n, y>0,y<m) и проверяет соседей данного элемента. И если сосед =1, то вызывает эту же функцию rek рекурсивно для соседа и помечает остров в массиве c[i,j]=true.

Алгоритм переборный, без рекурсии (неэффективный):

1. Объявляем переменную-метку label. Присваиваем ей значение 2.
2. Объявляем переменную логического типа flag – для отслеживания увеличения размера текущего «острова».
3. Идем в цикле do-while или repeat-until, пока флаг=true
4. Внутри цикла flag=false
5. Перебираем строки и столбцы, находим первую 1,
ставим на ее место label и flag=true.
6. Объявляем переменную логического типа flag_label. Создаем цикл while (flag_label), в котором еще два цикла for, в которых перебираем строки и столбцы.
Для каждого элемента a[i,j] проверяем:
если a[i,j]=1 и один из его соседей=label, то a[i,j]=label, flag_label=true.
7. Если внутри циклов for мы все перебрали, остров закончился – изменений не было, то flag_label будет false, тогда мы выйдем из цикла while и увеличивается значение label на 1.
8. Когда после прохождения циклов flag остался false, то все острова найдены и цикл останавливается.
9. В конце нужно вернуть значение label-1, это и есть искомое количество связных «островов» из единичек.

Приведем пример реализации эффективного алгоритма.

Пример программы на языке C++ (с векторами):

```
#include<bits/stdc++.h>
using namespace std;

void rek(long long x, long long y, long long n, long long m, vector<vector<long long>> &a, vector<vector<bool>> &c) {

    c[x][y] = true;
    long long v = 1;
    if (x>0)
        if (a[x - 1][y] == 1 and !c[x - 1][y]) {
            rek(x - 1, y, n, m, a, c);
        }
    if (y>0)
        if (a[x][y - 1] == 1 and !c[x][y - 1]) {
            rek(x, y - 1, n, m, a, c);
        }
    if (x<n-1)
        if (a[x + 1][y] == 1 and !c[x + 1][y]) {
            rek(x + 1, y, n, m, a, c);
        }
    if (y<m-1)
        if (a[x][y + 1] == 1 and !c[x][y + 1]) {
            rek(x, y + 1, n, m, a, c);
        }

}
```

```
int main() {
    long long n, m;
    cin >> n >> m;
    vector<vector<long long>> a;
    vector<vector<bool>> c;
    a.resize(n, vector<long long>(m , 0));
    c.resize(n, vector<bool>(m, false));
    for (long long i = 0; i < n; i++) {
        for (long long j = 0; j < m; j++) {
            cin >> a[i][j];
        }
    }
    long long kol = 0;
    for (long long i = 0; i < n; i++) {
        for (long long j = 0; j < m; j++) {
            if (!c[i][j] and a[i][j] == 1) {
                kol++;
               rek(i, j, n, m, a, c);
            }
        }
    }
    cout << kol;
    a.clear();
    c.clear();
    return 0;
}
```

Пример программы на языке Python:

```
n, m = list(map(int, input().split()))
a = []
for i in range(n):
    a.append(list(map(int, input().split())))

soz = 0
def rek(x, y, a):
    if a[x][y] == 1:
        a[x][y] = 2
    if x - 1 >= 0:
        if a[x - 1][y] == 1:
            a[x - 1][y] = 2
            a = rek(x - 1, y, a)
    if x + 1 < len(a):
        if a[x + 1][y] == 1:
            a[x + 1][y] = 2
            a = rek(x + 1, y, a)
    if y - 1 >= 0:
        if a[x][y - 1] == 1:
            a[x][y - 1] = 2
            a = rek(x, y - 1, a)
    if y + 1 < len(a[0]):
        if a[x][y + 1] == 1:
            a[x][y + 1] = 2
            a = rek(x, y + 1, a)
```

```
    return a


for i in range(len(a)):
    for j in range(len(a[0])):
        if a[i][j] == 1:
            soz += 1
            a = rek(i, j, a)
print(soz)
```

Допускаются и другие варианты решения задачи, приводящие к тем же результатам.

3 Вариант

№	Ответ	Балл	Решение
1.	30 и 38	20	Рассмотрим остатки при делении числа 53 на 11 и 17, это 9 и 2. Неизвестное нам число x на предыдущем шаге представлялось как x=y*11+z и x=p*17+q. Если z>q, то 53= x+z= y*11+z+z= y*11+2*z. Если z<q, то 53= x+q= y*17+q+q= y*17+2*q. Другими словами, один из этих остатков удвоили. Остаток 9 по модулю 11 всё равно, что 20. 1) Так как 20 является четным числом, то возможно удваивался первый остаток z (по модулю 11). Если это так, то получается, что начальный остаток z=10. Так как 53= x+z=x+10, то x=43. К полученному числу 43 нужно ещё раз применить этот же трюк: неизвестное нам число x на первом шаге также представлялось как x=y*11+z и x=p*17+q. Если z>q, то 43= x+z= y*11+2*z. Если z<q, то 43= x+q= y*17+2*q. Остатки при делении числа 43 на 11 и 17, это 10 и 9. Остаток 9 по модулю 17 всё равно, что 26. 1.1) Первый остаток равен 10, возможно этот остаток удваивался. Получается, что начальный остаток z=5. Так как 43= x+z=x+5, значит x=38. 1.2) Второй остаток 26, возможно этот остаток удваивался. Получается, что начальный остаток q=13. Так как 43= x+q=x+13, значит x=30. Таким образом, мы определили, что первоначальными числами в первом случае могли быть 30 или 38. 2) Второй остаток – при делении числа 53 на 17 – был равен 2. Если этот остаток удваивался, то получается, что начальный остаток q=1. Так как 53= x+q=x+1, то x=52. К полученному числу 52 снова применяем действия по нахождению остатков: если z>q, то 52= x+z=y*11+2*z. Если z<q, то 52= x+q= y*17+2*q. Остатки при делении числа 52 на 11 и 17, это 8 и 1. Остаток 1 по модулю 17 всё равно, что 18. 2.1) Первый остаток равен 8, значит, именно этот остаток удваивался. Получается, что начальный остаток z=4. Так как 52= x+z=x+4, значит x=48. Но для 48 не выполняется условие, что z>q, так как 48=4*11+4 и 48=2*17+14. А остаток 8 по модулю 11 всё равно, что 19 - нечётное число, не подходит.

<table>
<tr><td></td><td></td><td></td><td>2.2) Второй остаток 18, возможно этот остаток удваивался. Получается, что начальный остаток q=9.
Так как 52= x+q=x+9, значит x=43. Но для 30 не выполняется условие, что z<q, так как 43=3*11+10 и 43=2*17+9.
Значит, такой ситуации, что второй остаток – при делении числа 53 на 17 – был равен 2, не могло быть. Остаток 2 по модулю 17 всё равно, что 19. Так что такой ситуации тоже не могло быть. Значит, число 53 невозможно получить при делении на 17.
Таким образом, мы определили, что первоначальными числами могли быть 30 или 38.
Ответ: 30 и 38</td></tr>
<tr><td>2.</td><td>Первый игрок, Бельчонок</td><td>15</td><td>По условиям игры, выигрывает игрок, после хода которого, сумма координат становится не менее 73. Выигрышная стратегия есть у Бельчонка. Он может первым ходом удвоить вторую координату и привести фигуру на клетку (16,4). При любых ответных ходах Ёжика Бельчонок может воспользоваться преимуществом первого хода и свести ситуацию к такой, в которой Ёжик попадает в позицию, когда сумма удвоенной одной координаты и другой = 72. При такой стартовой позиции такая возможность гарантирована. То есть победит Бельчонок. Продемонстрируем выигрышную стратегию Бельчонка при любых ответных ходах противника. Обозначим Б – Бельчонок, Е – Ёжик
Обозначим ПБ – победа Бельчонка, ПЕ – победа Ёжика. Красным цветом будем помечать проигрышные для Бельчонка ситуации

<table>
<tr><th>Ход Б</th><th>Ход Е</th><th>Ход Б</th><th>Ход Е</th><th>Ход Б</th><th>Ход Е</th><th>Ход Б</th></tr>
<tr><td rowspan="14">16,4</td><td rowspan="5">32,4</td><td>32,8</td><td>33,8</td><td>66,8 ПБ</td><td></td><td></td></tr>
<tr><td></td><td>32,9</td><td>64,9 ПБ</td><td></td><td></td></tr>
<tr><td></td><td>64,8</td><td>128,8 ПБ</td><td></td><td></td></tr>
<tr><td></td><td>32,16</td><td>64,16 ПБ</td><td></td><td></td></tr>
<tr><td colspan="2">В остальных – ПЕ</td><td></td><td></td><td></td></tr>
<tr><td rowspan="5">16,8</td><td rowspan="4">32,8</td><td>33,8</td><td>66,8 ПБ</td><td></td><td></td></tr>
<tr><td>32,9</td><td>64,9 ПБ</td><td></td><td></td></tr>
<tr><td>64,8</td><td>128,8 ПБ</td><td></td><td></td></tr>
<tr><td>32,16</td><td>64,16 ПБ</td><td></td><td></td></tr>
<tr><td colspan="2">В остальных – ПЕ</td><td></td><td></td><td></td></tr>
<tr><td rowspan="3">17,4</td><td rowspan="3">34,4</td><td>35,4</td><td>70,6 ПБ</td><td></td><td></td></tr>
<tr><td>34,5</td><td>68,5 ПБ</td><td></td><td></td></tr>
<tr><td>68,4</td><td>136,4 ПБ</td><td></td><td></td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td>
<table>
<tr><td></td><td></td><td>34,8</td><td>68,8
ПБ</td><td></td><td></td></tr>
<tr><td></td><td colspan="2">В остальных – ПЕ</td><td></td><td></td><td></td></tr>
<tr><td></td><td rowspan="14">16,5</td><td rowspan="14">32,5</td><td rowspan="5">32,6</td><td rowspan="4">33,6</td><td>34,6</td><td>68,6
ПБ</td></tr>
<tr><td>33,7</td><td>66,7
ПБ</td></tr>
<tr><td>66,6</td><td>132,6
ПБ</td></tr>
<tr><td>33,12</td><td>66,12
ПБ</td></tr>
<tr><td colspan="2">В остальных – ПЕ</td><td></td></tr>
<tr><td rowspan="5">33,5</td><td rowspan="4">33,6</td><td>34,6</td><td>68,6
ПБ</td></tr>
<tr><td>33,7</td><td>66,7
ПБ</td></tr>
<tr><td>66,6</td><td>132,6
ПБ</td></tr>
<tr><td>33,12</td><td>66,12
ПБ</td></tr>
<tr><td colspan="2">В остальных – ПЕ</td><td></td></tr>
<tr><td>64,5</td><td>128,5
ПБ</td><td></td><td></td></tr>
<tr><td>32,10</td><td>64,10
ПБ</td><td></td><td></td></tr>
<tr><td colspan="2">В остальных – ПЕ</td><td></td><td></td><td></td></tr>
</table>

В остальных случаях (при других первых ходах Бельчонка) существуют целые ветки с выигрышами Ёжика. Например, при первом ходе Бельчонка (16,2)->(32,2). Ёжик может ответить (32,3). При любом ответе Бельчонка – (64,3), (32,6), (33,3), (34,4) Ёжик победит, так как сможет свести ситуацию к позицию, когда сумма удвоенной одной координаты и другой = 72 для Бельчонка!
Ответ: Победит Бельчонок (первый игрок), его первый ход (16,2)->(16,4). Выигрышная стратегия указана в таблице.</td></tr>
<tr><td>3.</td><td>402</td><td>10</td><td>Из условия задачи следует, что нам нужно найти максимальное количество ресурсов, которое может собрать Бельчонок при заданных ограничениях, пройдя из левой нижней клетки в правую верхнюю. Причём движение начинается из второго столбца, из ячейки B14, поэтому столбцом A можно пренебречь.
В условиях задачи имеются два ограничения:
1) карта эффектов увеличивает на два количество ресурсов в клетках с нечётными значениями.</td></tr>
</table>

			2) с каждой клетки Бельчонок может собрать максимум 15 единиц ресурсов, т.е. если в ячейке находится число больше 15, то взять из нее мы можем только 15. Таким образом, перед началом движения нужно первым действием прибавить 2 к значениям в ячейках, в которых содержатся нечётные числа. После этого вторым действием нужно проставить в каждую ячейку со значением >15, значение 15. Оптимальнее всего объединить эти два действия в одно. Таким образом, для решения этой задачи можно пользоваться двумя способами. **Первый способ (менее эффективный):** 1) Исходные данные находятся в области B1:O14. Из исходной таблицы получим измененную с помощью формулы =ЕСЛИ(ОСТАТ(B1;2)=1;A1+2;A1) – это делаем для каждой ячейки. Здесь мы сразу прибавляем 2 для всех нечётных чисел (с остатком 1 при делении на 2). Пусть измененные значения находятся в ячейках B16:O29. 2) Из предыдущей таблицы получим еще одну измененную с помощью формулы =ЕСЛИ(B16<15;B16;15) – это делаем для каждой ячейки. Здесь мы все значения > 15 заменяем на 15. Пусть полученные измененные значения находятся в ячейках B31:O44. 2) Теперь будем искать максимальное значение. Для поиска максимального значения будем работать с областью B46:O59, при расчетах будем использовать значения в вышеполученной области B31:O44. В ячейку B59 напишем значение **=B46**. Для каждой ячейки левого столбца это будет сумма всех ячеек ниже от текущей. Внесем в ячейку B58 формулу **=B43+B59** и скопируем за маркер вверх до ячейки B46. Для каждой ячейки строки №59 это будет сумма всех ячеек левее от текущей. Внесем в ячейку C59 формулу **=C44+B59** и скопируем за маркер вправо до ячейки O59. Далее в ячейку C58 вставим формулу **= =C43+МАКС(B58;C59).** Т.е. мы суммируем значение, которое находится в данной ячейке, и максимальное число, полученное на предыдущем шаге (придя из соседних по движению, так как в каждую ячейку можно прийти либо слева, либо снизу). Скопируем эту формулу, потянув за маркер в ячейки C58:O46. Значение в ячейке O46 будет максимальным количеством ресурсов, которое может собрать Бельчонок — 402. **Второй способ (эффективный):** В этом алгоритме объединяются пункты 1) и 2) предыдущего алгоритма. Вместо двух промежуточных таблиц получаем только одну с помощью формулы: =ЕСЛИ(B1>15;15;ЕСЛИ(ОСТАТ(B1;2)=1;B1+2;B1)) – это делаем для каждой ячейки. Здесь мы сразу все значения > 15 заменяем на 15 и прибавляем 2 к нечётным числам<15. При прибавлении 2 к нечетным числам>15 (17,19,...) получаемые числа всё равно >15, а значит, они будут просто заменены на 15. Здесь ситуация ничем не отличается от случая с чётными числами, т.е. для любых четных и нечётных чисел >15 можно просто выполнить только одно действие – замену на 15. Осталось проверить, не будет ли нарушения условий, если у нас произойдет прибавления 2 к максимальному нечетному числу<15. Это число 13. 13+2=15. Условие не нарушено. Все остальные действия алгоритма по поиску максимального значения схожи с первым алгоритмом.

			Ответ: 402
4.	Написанная программа	25	Нужно найти в массиве (*или векторе, списке*) два таких числа, удаление которых из массива минимально изменяет среднее значение новой полученной последовательности. В качестве ответа нужно вернуть сумму этих значений. На первый взгляд, кажется, что эта задача решается удалением из массива двух чисел наиболее близко лежащими к среднему значению, или, наоборот, двух наиболее отдаленных элемента. Но, к сожалению, это не так. Может оказаться, что наиболее сильно влияет пара элементов, находящихся в произвольных местах массива. Приведем пример такого массива: 1 2 3 30 70 100 Среднее=34.33 Если удалить 1 и 2, то среднее=50.75 Если удалить 3 и 30, то среднее=43.25 Если удалить 70 и 100, то среднее=9 Если удалить 1 и 70, то среднее=33.75. Оно ближе всего к 34.33. Данную задачу можно решить несколькими способами. Рассмотрим самый оптимальный из них. **Способ решения задачи**: mas – массив элементов sum – сумма всех элементов массива x, y – пара элементов, наименее изменяющие среднее sred – текущее среднее значение (без пары x, y) minsred – минимальное среднее, i, j – счётчики. 1. Считываем число N. 2. Инициализируем переменную sum=0; 3. В цикле for от 1 до N выполняем считывание элементов массива mas[i]. (Если позволяет язык программирования, то сразу же накапливаем сумму элементов массива: sum=sum+mas[i].) 4. Если суммирование не выполнялось при считывании, то выполняем его сейчас. 5. Присваиваем x=mas[0], y=mas[1], minsred=(sum-x-y)/(n-2). 6. Во вложенных циклах перебираем все возможные пары чисел. Первый цикл по i перебирает все элементы от 1 до предпоследнего (i – индекс элемента массива, который будет первым в паре). Второй по j перебирает все элементы от следующего за текущим (т.е. j=i+1) до последнего (j – индекс элемента массива, который будут вторым в паре). 3.1) находим сумму текущее среднее по формуле sred =(sum-mas[i] + mas[j])/(n-2). 3.2) Выполняем проверку среднего sred, что оно дает наименьшее отклонение от первоначального среднего, чем хранимое минимальное (в переменной minsred). Т.е. сравниваем abs(sum/n- sred)<abs(sum/n-minsred). Если так, то запоминаем x=mas[i], y=mas[j], minsred=sred. 4) Повторяем действия для следующих пар.

5) В полученных значениях переменных x y будет содержаться пара, удаление которых из массива минимально изменяет среднее значение новой последовательности. Выводим сумму значений x+y на экран.
Использование векторов или списков при решении данной задачи является еще более эффективным способом решения задачи.
Допускаются и другие варианты решения задачи, приводящие к тем же результатам.
Другие алгоритмы решения данной задачи (с отниманием и возвратом элементов, с накоплением разниц в других массивах и т.д.) не являются эффективными.

Пример программы на языке C++:

```
#include <iostream>
using namespace std;

int main()
{
   long long n, x, y, sum,sred,minsred;
   int i,j;
   cin >> n;
   int mas[10000];
   for (i = 0; i < n; i++) {
      cin >> mas[i];
      sum += mas[i];
   }
   x=mas[0];
   y=mas[1];
   minsred=(sum-x-y)/(n-2);
   for (i = 0; i < n-1; i++)
      for (j = i+1; j < n; j++) {
         sred=(sum-mas[i]-mas[j])/(n-2);
         if (abs(sum/n- sred)<abs(sum/n-minsred))
         {
            minsred=sred;
            x=mas[i];
            y=mas[j];
         }
   }
   cout << x+y;
   return 0;
}
```

Пример программы на языке C++ (с векторами):

```
#include <iostream>
#include <vector>
using namespace std;
```

```
int main()
{
   long long n, x, y, sum,sred,minsred;
   int i,j;
   cin >> n;
   vector <long long>mas(n);
   for (i = 0; i < n; i++) {
      cin >> mas[i];
      sum += mas[i];
   }
   x=mas[0];
   y=mas[1];
   minsred=(sum-x-y)/(n-2);
   for (i = 0; i < n-1; i++)
      for (j = i+1; j < n; j++) {
         sred=(sum-mas[i]-mas[j])/(n-2);
         if (abs(sum/n- sred)<abs(sum/n-minsred))
         {
            minsred=sred;
            x=mas[i];
            y=mas[j];
         }
   }
   cout << x+y;
   return 0;
}
```

Пример программы на языке Python:

```
n=int(input())
mas=list(map(int, input().split()))
summ=sum(mas)
x=mas[0]
y=mas[1]
minsred=(summ-x-y)/(n-2)
for i in range(n-1):
   for j in range(i+1,n):
      sred=(summ-mas[i]-mas[j])/(n-2)
      if abs(summ/n- sred)<abs(summ/n-minsred):
         minsred=sred
         x=mas[i]
         y=mas[j]
print(x+y)
```

5.		30	Алгоритм решения задачи совпадает с решением задачи в варианте 1.

4 вариант

	Ответ	Балл	Решение
1.	32 и 33	20	Рассмотрим остатки при делении числа 50 на 13 и 15, это 11 и 5. Неизвестное нам число x на предыдущем шаге представлялось как x=y*13+z и x=p*15+q. Если z>q, то 50= x+z= y*13+z+z= y*13+2*z. Если z<q, то 50= x+q= y*15+q+q= y*15+2*q. Другими словами, один из этих остатков удвоили. Остаток 11 по модулю 13 всё равно, что 24, а 5 по модулю 15 всё равно, что 20. 1) Так как 24 является четным числом, то возможно удваивался первый остаток z (по модулю 13). Если это так, то получается, что начальный остаток z=12. Так как 50= x+z=x+12, то x=38. К полученному числу 38 нужно ещё раз применить этот же трюк: неизвестное нам число x на первом шаге также представлялось как x=y*13+z и x=p*15+q. Если z>q, то 38= x+z= y*13+2*z. Если z<q, то 38= x+q= y*15+2*q. Остатки при делении числа 38 на 13 и 15, это 12 и 8. 1.1) Первый остаток равен 12, возможно именно этот остаток удваивался. Получается, что начальный остаток z=6. Так как 38= x+z=x+6, значит x=32. 1.2) Второй остаток 8, возможно этот остаток удваивался. Получается, что начальный остаток q=4. Так как 38= x+q=x+4, значит x=34. Но для 34 не выполняется условие, что z<q, так как 34=2*13+8 и 34=2*15+4. Т.е. должно было прибавиться 8, а не 4! А остаток 8 по модулю 15 всё равно, что 23 - нечётное число, не подходит. Таким образом, мы определили, что первоначальным числом в первом случае могло быть только число 32. 2) Второй остаток при делении числа 50 на 15 был равен 20 (по модулю 15). Если этот остаток удваивался, то получается, что начальный остаток q=10. Так как 50= x+q=x+10, то x=40. К полученному числу 40 снова применяем действия по нахождению остатков: если z>q, то 40= x+z=y*13+2*z. Если z<q, то 40= x+q= y*15+2*q. Остатки при делении числа 40 на 13 и 15, это 1 и 10. Остаток 1 по модулю 13 всё равно, что 14. 2.1) Первый остаток равен 14, значит, этот остаток мог удваиваться. Получается, что начальный остаток z=7. Так как 40= x+z=x+7, значит x=33. 2.2) Второй остаток 10, возможно этот остаток удваивался. Получается, что начальный остаток q=5. Так как 40= x+q=x+5, значит x=35. Но для 35 не выполняется условие, что z<q, так как 35=2*13+9 и 35=2*15+5. Т.е. должно было прибавиться 9, а не 5! Значит, число 35 нам не подходит. Таким образом, мы определили, что первоначальным числом во втором случае могло быть 33. Ответ: 32 и 33.
2.	Первый игрок Бельчонок	15	По условиям игры, выигрывает игрок, после хода которого сумма координат становится не менее 32. Выигрышная стратегия есть у Бельчонка. Он может первым ходом удвоить первую координату и привести фигуру на клетку (10,3). При любых ответных ходах Ёжика Бельчонок может воспользоваться преимуществом первого хода и свести ситуацию к такой, в которой Ёжик попадает в позицию, когда сумма удвоенной одной координаты и другой = 31.

При такой стартовой позиции такая возможность гарантирована. То есть победит Бельчонок. Продемонстрируем выигрышную стратегию Бельчонка при любых ответных ходах противника. Обозначим Б – Бельчонок, Е – Ёжик
Обозначим ПБ – победа Бельчонка, ПЕ – победа Ёжика. Красным цветом будем помечать проигрышные для Бельчонка ситуации

<table>
<tr><th>Ход Б</th><th>Ход Е</th><th>Ход Б</th><th>Ход Е</th><th>Ход Б</th></tr>
<tr><td rowspan="14">10,3</td><td>20,3</td><td>40,3
ПБ</td><td></td><td></td></tr>
<tr><td rowspan="6">10,5</td><td rowspan="4">10,10</td><td>12,10</td><td>24,10
ПБ</td></tr>
<tr><td>10,12</td><td>10,24
ПБ</td></tr>
<tr><td>20,10</td><td>40,10
ПБ</td></tr>
<tr><td>10,20</td><td>10,40
ПБ</td></tr>
<tr><td colspan="2">В остальных – ПЕ</td><td></td></tr>
<tr><td rowspan="6">10,6</td><td rowspan="4">12,6</td><td>12,8</td><td>24,8
ПБ</td></tr>
<tr><td>14,6</td><td>28,6
ПБ</td></tr>
<tr><td>24,6</td><td>48,6
ПБ</td></tr>
<tr><td>12,12</td><td>24,12
ПБ</td></tr>
<tr><td colspan="2">В остальных – ПЕ</td><td></td></tr>
<tr><td rowspan="6">12,3</td><td rowspan="4">12,6</td><td>12,8</td><td>24,8
ПБ</td></tr>
<tr><td>14,6</td><td>28,6
ПБ</td></tr>
<tr><td>24,6</td><td>48,6
ПБ</td></tr>
<tr><td>12,12</td><td>24,12
ПБ</td></tr>
<tr><td colspan="2">В остальных – ПЕ</td><td></td></tr>
</table>

В остальных случаях (при других первых ходах Бельчонка) существуют целые ветки с выигрышами Ёжика.
Ответ: Победит Бельчонок (первый игрок), его первый ход (5,3)->(10,3). Выигрышная стратегия указана в таблице.

3.	322	10	Из условия задачи следует, что нам нужно найти максимальное количество ресурсов, которое может собрать Бельчонок при заданных ограничениях, пройдя из **правой верхней клетки** в **левую нижнюю**. Причём движение начинается из третьего с края столбца, из ячейки N1, поэтому столбцами O и P можно пренебречь. В условиях задачи имеются два ограничения: 1) карта эффектов увеличивает на 2 количество ресурсов в клетках, находящихся в четных строках (т.е. номер строки является четным числом). 2) с каждой клетки Бельчонок может собрать максимум 12 единиц ресурсов, т.е. если в ячейке находится число больше 12, то взять из нее мы можем только 12. Таким образом, перед началом движения нужно первым действием прибавить 2 к значениям в ячейках, находящихся в четных строках. После этого вторым действием нужно проставить в каждую ячейку со значением >12, значение 12. Оптимальнее всего объединить эти два действия в одно. Таким образом, для решения этой задачи можно пользоваться двумя способами. **Первый способ (менее эффективный):** 1) Исходные данные находятся в области A1:N14. Из исходной таблицы получим измененную с помощью формулы =ЕСЛИ(ОСТАТ(СТРОКА(A1);2)=0;A1+2;A1) – это делаем для каждой ячейки. Здесь мы сразу прибавляем 2 к значениям в ячейках, находящихся в четных строках. Пусть измененные значения находятся в ячейках A16:N29. 2) Из предыдущей таблицы получим еще одну измененную с помощью формулы =ЕСЛИ(A16<12;A16;12) – это делаем для каждой ячейки. Здесь мы все значения > 12 заменяем на 12. Пусть полученные измененные значения находятся в ячейках A31:N44. 2) Теперь будем искать максимальное значение. Для поиска максимального значения будем работать с областью A46:N59, при расчетах будем использовать значения в вышеполученной области A31:N44. В ячейку N46 напишем значение **=N31**. Для каждой ячейки правого столбца это будет сумма всех ячеек выше от текущей. Внесем в ячейку N47 формулу **=N32+N46** и скопируем за маркер вниз до ячейки N59. Для каждой ячейки строки №46 это будет сумма всех ячеек правее от текущей. Внесем в ячейку M46 формулу **=M31+N46** и скопируем за маркер влево до ячейки A46. Далее в ячейку M47 вставим формулу **=M32+МАКС(M46;N47).** Т.е. мы суммируем значение, которое находится в данной ячейке, и максимальное число, полученное на предыдущем шаге (придя из соседних по движению, так как в каждую ячейку можно прийти либо справа, либо сверху). Скопируем эту формулу, потянув за маркер в ячейки A59:M47. Значение в ячейке A59 будет максимальным количеством ресурсов, которое может собрать Бельчонок — 322. **Второй способ (эффективный):** В этом алгоритме объединяются пункты 1) и 2) предыдущего алгоритма. Вместо двух промежуточных таблиц получаем только одну с помощью формулы: =МИН(12;ЕСЛИ(ОСТАТ(СТРОКА(A1);2)=0;A1+2;A1)) В этой формуле вначале проверяется, что если строка четная, то мы прибавляем 2 к значению. Если это не так, то значение не изменяется. И потом результат сравнивается с числом 12. Если наше значение >12, то оно будет заменено на 12. Если это не так, то значение не изменяется.

			Все остальные действия алгоритма по поиску максимального значения схожи с первым алгоритмом. Ответ: 322

4.		25	или максимумов, то, что даст максимальное отклонение от начального среднего – эффективный алгоритм. **Первый алгоритм (переборный, неэффективный)**: mas – массив элементов sum – сумма всех элементов массива x, y – пара элементов, наиболее изменяющие среднее, sred – текущее среднее значение (без пары x, y) maxsred – максильное среднее, i, j – счётчики. 1. Считываем число N. 2. Инициализируем переменную sum=0; 3. В цикле for от 1 до N выполняем считывание элементов массива mas[i]. (Если позволяет язык программирования, то сразу же накапливаем сумму элементов массива: sum=sum+mas[i].) 4. Если суммирование не выполнялось при считывании, то выполняем его сейчас. 5. Присваиваем x=mas[0], y=mas[1], maxsred=(sum-x-y)/(n-2). 6. Во вложенных циклах перебираем все возможные пары чисел. Первый цикл по i перебирает все элементы от 1 до предпоследнего (i – индекс элемента массива, который будет первым в паре). Второй по j перебирает все элементы от следующего за текущим (т.е. j=i+1) до последнего (j – индекс элемента массива, который будут вторым в паре). 3.1) находим сумму текущее среднее по формуле sred =(sum-mas[i] + mas[j])/(n-2). 3.2) Выполняем проверку среднего sred, что оно дает наибольшее отклонение от первоначального среднего, чем хранимое максимальное (в переменной maxsred). Т.е. сравниваем abs(sum/n- sred)>abs(sum/n-maxsred). Если так, то запоминаем x=mas[i], y=mas[j], maxsred=sred. 4) Повторяем действия для следующих пар. 5) В полученных значениях переменных x y будет содержаться пара, удаление которых из массива максимально изменяет среднее значение новой последовательности. 6) Если сумма x+y является четной, то вывести на экран x+y. В противном случае вернем 0. Второй алгоритм (оптимальный) mas – массив элементов sum – сумма всех элементов массива max1, max2 – два максимальных элемента, min1, min2 – два минимальных элемента srMIN – среднее без минимумов, srMAX – среднее без минимумов, x, y – пара элементов, наиболее изменяющие среднее i – счётчик. 1. Считываем число N.

2. Инициализируем переменную sum=0; max1 = 0, max2 = 0, min1 = 100000000, min2 = 100000000;
3. В цикле for от 1 до N выполняем считывание элементов массива mas[i]. (Если позволяет язык программирования, то сразу же накапливаем сумму элементов массива: sum=sum+mas[i] и ищем минимумы и максимумы. Если язык не позволяет это сделать, значит после считывания будет еще один цикл, в котором выполняется суммирование и поиск двух максимумов и двух минимумов.
3.1) поиск максимума: для текущего элемента mas[i] выполняем сравнение если max1 < mas[i], то
max2 = max1 и max1 = mas[i]. Если это не так, то наше число меньше первого максимума, но возможно оно является вторым по значению максимумом?
Проверяем это: если max2 < mas[i], то max2 = mas[i].
3.2) Схожим образом выполняется поиск двух минимумов: для текущего элемента mas[i] выполняем сравнение если min1 > mas[i], то min2 = min1 и min1 = mas[i]. Если это не так, и элемент mas[i] больше первого минимума, но возможно он является вторым по значению минимумом?
Проверяем это: если(min2 > mas[i]), то min2 = mas[i].
4. После выхода из цикла мы знаем sum, max1,max2,min1,min2.
5. Найдем среднее без минимумов srMIN = (sum - min1 - min2) / (n - 2);
6. Найдем среднее без максимумов srMAX = (sum - max1 - max2) / (n - 2);
7. Выбрать из srMIN и srMAX то, что даст максимальное отклонение от начального среднего
если (abs(sum/n - srMIN) > abs(sum/n - srMAX)), то
x=min1, y=min2,
иначе x=max2, ymax1.
8. Если сумма x+y является четной, то вывести на экран x+y. В противном случае вернем 0.

Использование векторов или списков при решении данной задачи является еще более эффективным способом решения задачи.

Допускаются и другие варианты решения задачи, приводящие к тем же результатам.
Другие алгоритмы решения данной задачи (с отниманием и возвратом элементов, с накоплением разниц в других массивах и т.д.) не являются эффективными.

Пример программы на языке C++:

```
#include <iostream>
using namespace std;

int main() {
   int n, max1 = 0, max2 = 0, min1 = 100000000, min2 = 100000000;
   double sum = 0;
```

```
    cin >> n;
    int mas[10000];
    for (int i = 0; i < n; ++i) {
        cin >> mas[i];
        sum += mas[i];
        if (max1 < mas[i]) {
            max2 = max1;
            max1 = mas[i];
        } else if (max2 < mas[i]) {
            max2 = mas[i];
        }
        if (min1 > mas[i]) {
            min2 = min1;
            min1 = mas[i];
        } else if (min2 > mas[i]) {
            min2 = mas[i];
        }
    }

    double srMIN, srMAX;
    int x=0, y=0;

    srMIN = (sum - min1 - min2) / (n - 2);
    srMAX = (sum - max1 - max2) / (n - 2);
    if (abs(sum/n - srMIN) > abs(sum/n - srMAX)) {
        x=min1;
        y=min2;
    }
    else {
            x=max2;
            y=max1;
    }

    if ((x + y) % 2 == 0)
      cout << x<< ' ' << y;
    else
     cout<<"0";
    return 0;
}
```

Пример программы на языке C++ (с векторами):

```
#include <iostream>
#include <vector>
using namespace std;
using namespace std;
```

```
int main() {
   int n, max1 = 0, max2 = 0, min1 = 100000000, min2 = 100000000;
   double sum = 0;
   cin >> n;
   vector<int> mas(n);
   for (int i = 0; i < n; ++i) {
      cin >> mas[i];
      sum += mas[i];
      if (max1 < mas[i]) {
         max2 = max1;
         max1 = mas[i];
      } else if (max2 < mas[i]) {
         max2 = mas[i];
      }
      if (min1 > mas[i]) {
         min2 = min1;
         min1 = mas[i];
      } else if (min2 > mas[i]) {
         min2 = mas[i];
      }
   }

   double srMIN, srMAX;
   int x=0, y=0;

   srMIN = (sum - min1 - min2) / (n - 2);
   srMAX = (sum - max1 - max2) / (n - 2);
   if (abs(sum/n - srMIN) > abs(sum/n - srMAX)) {
      x=min1;
      y=min2;
   }
   else {
         x=max2;
         y=max1;
   }

   if ((x + y) % 2 == 0)
     cout << x<< ' ' << y;
   else
    cout<<"0";
   return 0;
}
```

Пример программы на языке Python:

```
n=int(input())
```

			mas=list(map(int, input().split()))
			summ=sum(mas)
			
			max1 = 0
			max2 = 0
			min1 = 100000000
			min2 = 100000000;
			
			for i in range(n):
			  if max1 < mas[i]:
			  max2 = max1
			  max1 = mas[i]
			  elif max2 < mas[i]:
			  max2 = mas[i]
			
			  if min1 > mas[i]:
			  min2 = min1
			  min1 = mas[i]
			  elif min2 > mas[i]:
			  min2 = mas[i]
			
			x=0
			y=0
			srMIN = (summ - min1 - min2) / (n - 2)
			srMAX = (summ - max1 - max2) / (n - 2)
			if abs(summ/n - srMIN) > abs(summ/n - srMAX):
			  x=min1
			  y=min2
			else:
			  x=max2
			  y=max1
			
			if (x + y) % 2 == 0:
			  print(x, y)
			else:
			  print(0)

5		30	Алгоритм решения задачи совпадает с решением задачи в варианта 1. Единственное отличие только в инвертировании – при решении задачи нужно заменить 0 на 1, и, наоборот, 1 на 0. Так как нам нужно найти максимальный связный «остров», соединенных между собой нулей.

Информатика. 10 класс
Критерии оценивания

Для всех вариантов

Задача 1.

Правильный ответ с полным объяснением – 20 баллов.
Верное рассуждение, ответ частично неверный вследствие арифметических или логических ошибок – за каждое несовпадение отнимается 3 балла.
Правильный ответ, решение подбором с проверками выполнения условий задачи – 16 баллов.
Верное рассуждение, но неполное, ответ верный – 15 баллов
Правильный ответ, решение подбором без проверок выполнения условий задачи – 14 баллов.
Частично верное рассуждение, найдено верно только одно значение из двух – 12 баллов
Решение подбором с проверками выполнения условий задачи, найдено верно только одно значение из двух – 12 баллов.
Ответ верный, нет объяснения, каким образом найдено значение, только проверяется выполнение условия задания для значения – 11 баллов
Сделана только половина задания, состоящего из двух частей (причем выполнено верно) – 10 баллов.
Найдено верно одно значение из двух, объяснение неполное – 10 баллов.
Программное решение вместо рассуждения, ответ верный – 10 баллов.
Найдено верно одно значение из двух, выполнены вычисления без пояснений – 9 баллов.
Программное решение вместо рассуждения, найдено верно только одно значение из двух – 9 баллов.
Верное рассуждение, ответ неверный – 8 баллов.
Частично верное рассуждение, ответ неверный – 7 баллов.
Частично верное рассуждение, ответа нет – 7 баллов.
Неверное рассуждение, неправильный ответ – 5 баллов.
Неверное рассуждение, ответа нет – 4 баллов
Правильный ответ без пояснения – 3 балла.
Другой ответ – 0 баллов.

Задача 2.

Правильный ответ с полным объяснением – 15 баллов.
Верное рассуждение, ответ неверный (из-за арифметических ошибок) – 12 баллов.
В целом верное рассуждение, разобраны не все варианты стратегий выигрыша – 12 баллов.
В целом верное рассуждение, но недостаточно подробное, ответ верный – 11 баллов.
Частично верное рассуждение, ответ получен верный – 9 баллов.
Частично верное рассуждение, ответ неверный – 7 баллов.
Частично верное рассуждение, ответа нет – 7 баллов.
Неполное рассуждение, стратегия не описана полно (или слишком расплывчато), ответ верен – 7 баллов.
Неверное рассуждение, ответ верный – 6 баллов.
Неверное рассуждение, неправильный ответ – 5 баллов.
Неверное рассуждение, ответа нет – 5 баллов.
Правильный ответ без пояснения – 3 балла.
Другой ответ – 0 баллов.

Задача 3.

Правильный ответ и предоставлен файл с электронной таблицей с верными расчётами с использование формул и функций– 10 баллов.

Предоставлен файл с электронной таблицей с расчётами, близкими к верному – отличие в одном действии (повлиявшем на результат) – 9 баллов
Правильный ответ и предоставлен файл с электронной таблицей с верными расчётами, но не доведенными до финальной стадии – 8 баллов
Неправильный ответ и предоставлен файл с электронной таблицей с верными расчётами – 8 баллов
Правильный ответ и предоставлен файл с электронной таблицей с верными расчётами, но без использования формул и функций – 7 баллов
Неправильный ответ и предоставлен файл с электронной таблицей с частично верными расчётами (не учтено одно из ограничений задачи) – 6 баллов.
Правильный ответ и предоставлен файл с электронной таблицей с неверными расчётами – 5 баллов.
Неправильный ответ и предоставлен файл с электронной таблицей с частично верными расчётами – 5 баллов.
Неправильный ответ и предоставлен файл с электронной таблицей с неверными расчётами – 4 балла.
Правильный ответ и не предоставлен файл с электронной таблицей, есть программа с верными расчётами – 4 балла.
Неправильный ответ и не предоставлен файл с электронной таблицей, есть программа с неверными расчётами – 3 балла.
Неправильный ответ и предоставлен файл с электронной таблицей без расчётов – 2 балла.
Правильный ответ и не предоставлен файл с электронной таблицей с расчётами – 2 балла.
Ответ близок к правильному и не предоставлен файл с электронной таблицей с расчётами – 1 балл.
Другой ответ – 0 баллов.
Решен не свой вариант, ответ и расчёты верные – 5 баллов.

Задача 4.
Правильно решающий задачу, работающий и эффективный программный код – 25 баллов.
Правильно решающий задачу, работающий и неэффективный программный код – 23 балла.
Работающий и эффективный программный код, но есть незначительные ошибки в работе алгоритма – 19 баллов
Есть некритичная ошибка в алгоритме, при ее исправлении алгоритм работает, и он будет эффективным – 17 баллов
Программный код работает, но он неэффективный и есть незначительные ошибки в работе алгоритма (частично верный код) – 15 баллов
Программный код частично верный, но не работающий – 12 баллов
Программный код работающий, но большая часть алгоритма ошибочна – 10 баллов
Программный код работающий, но полностью ошибочный – 8 баллов
Программный код неверный и не работающий – 5 баллов
Программный код неверный и не дописан – 4 баллов
Описан алгоритм работы программы, но не написан программный код – 2 балла.
Другой ответ – 0 баллов.

Задача 5.
Правильно решающий задачу, работающий и эффективный программный код – 30 баллов.
Правильно решающий задачу, работающий и неэффективный программный код – 28 баллов.
Работающий и эффективный программный код, есть незначительные ошибки в работе алгоритма – 25 баллов.
Работающий и неэффективный программный код, есть незначительные ошибки в работе алгоритма – 22 балла

Есть некритичная ошибка в алгоритме, при ее исправлении алгоритм работает, и он будет эффективным – 19 баллов
Программный код работает, частично верный код, есть значительные ошибки в работе алгоритма – 15 баллов
Программный код частично верный, но не работающий – 15 баллов
Программный код работающий, но полностью ошибочный – 10 баллов
Программный код работающий, но решающий другую задачу (включая другой вариант) – 9 баллов
Программный код неверный и не работающий – 6 баллов
Программный код не дописан – 4 балла
Описан алгоритм работы программы, но не написан программный код – 3 балла.
Другой ответ – 0 баллов

Информатика. 11 класс
Решения

1 вариант

№	Правильный ответ	Балл
1.	Во второй строчке таблицы если x → ¬y = 0, то x=y=1. Значит, оставшийся ноль может быть только z, следовательно, первая переменная z. Далее, если y ∧ ¬z = 0, то либо y=0, либо z=1, но в третьей строчке z = 0, значит, в оставшемся пустом месте для второй переменной должен быть ноль, и вторая переменная y. Итого zyx. Теперь если вся функция G равна нуля, то каждое из F должно быть равно нулю. Из второй и 4 строчки получаем F(1, 1) = F(1, 0) = F(0, 1) = 0. Если бы F(0,0) был равен 0, то G всегда была бы 0, следовательно, F(0, 0) = 1 и F(x, y) = ¬x and ¬y	16
2.	1) От кучи 97 отнимаем три спички, получаем позицию (94, 94), далее ходим симметрично, побеждает первый игрок. Перед следующими задачами стоит убедиться (например, с помощью обычного перебора), что если игра начинается с одной кучи и в куче от 3 до 10 спичек, то проигрышные позиции только 6 и 10. 2) Разобьём игру на две параллельные, одна с кучей из 6 спичек, другая из 27. Легко заметить, например, с помощью дерева, что любой ход с кучей в 6 спичек приводит к моментальному выигрышу второго игрока, то есть это проигрышная позиция. Из кучи в 27 спичек можно отнять одну спичку и поделить на 2 кучи по 13. Получается, что Ада перевела Настю в позицию, где две параллельные игры проигрышные. Это тоже, очевидно, проигрышная позиция, потому что любой ход приводит к комбинации проигрышной и выигрышной позиции, а из такой комбинации можно перейти снова в позицию минус-минус. Если Настя походит в одной из куч по 13, Ада сделает симметричный ход. Если Настя походит в куче 6, то Ада сразу же закончит эту кучу. Побеждает Ада. 3) Если Ада как-то походит в куче 3, то любой её ход сразу же приведёт к позиции (8) (возможно, с двумя кучами по одной спичке, но они уже роли не играют). Тогда Настя ходит в (6, 1) и выигрывает (6 позиция проигрышная). Значит, у Ады есть ходы (5,3), (6, 1, 3), (5, 2, 3), (4, 3, 3). В (4, 3, 3) остаётся ровно 3 хода, значит, Настя походит последней в любом случае, в (6, 1, 3) можно любым способом убрать кучу из 3 спичек и получить позицию, эквивалентную позиции из 6 спичек (проигрышной), в (5, 3) или (5, 2, 3) можно разбить пятерку на 3 и 1 и получить позицию, эквивалентную (3, 4), в которой осталось ровно два хода и последний ход за Настей. Настя выигрывает. 4) Ада первым ходом ходит в позицию 9, 4 и выигрывает. Действительно, 9 выигрышная позиция (можно попасть в 6, проигрышную). Любой ход в куче 4 уничтожит эту кучу, что поставит Аду в выигрышную позицию. Если же Настя походит из 9 в 6 или (6,2), то получится комбинация выигрышной плюс проигрышной позиции для Ады. Если Настя походит в (7, 1, 4), Ада походит в (4, 2, 1, 4), где остаётся ровно два хода. Если Настя походит в (3, 5, 4), Ада походит в (3, 2, 2, 1, 4), где остаётся ровно два хода. Ада выигрывает.	20
3.	539756. Вычислять лучше всего с помощью Excel, выделив все случаи, когда робот может взять ровно 3 монеты и сымитировав их в таблице.	18
4.	M = int(input())	21

	FA = [[-1 for i in range(5)] for j in range(3*M)]
	def s(n):
	  k = 0
	  while n:
	  k += n%5
	  n //= 5
	  return k
	def f(n, m, op = 0):
	  if n>m: return 0
	  if n==m: return 1
	  ans = 0
	  if op != 1:
	  if FA[n+1][1] == -1:
	  FA[n+1][1] = f(n+1, m, 1)
	  ans += FA[n+1][1]
	  if op != 2:
	  if FA[n+2][2] == -1:
	  FA[n+2][2] = f(n+2, m, 2)
	  ans += FA[n+2][2]
	  if op != 3:
	  if FA[n+3][3] == -1:
	  FA[n+3][3] = f(n+3, m, 3)
	  ans += FA[n+3][3]
	  if op != 4:
	  if FA[n+s(n)][4] == -1:
	  FA[n+s(n)][4] = f(n+s(n), m, 4)
	  ans += FA[n+s(n)][4]
	  return ans
	print(f(1, M))
	Стоит обратить внимание, что обычное динамическое программирование будет работать очень долго на некоторых тестах. Стоит провести запись значений в специальный массив.	

5.	N = int(input())
	A = map(int, input().split())
	Aminus0, Aminus1, Aplus0, Aplus1 = [], [], [], []
	for i in A:
	  if i >= 0 and i %5 != 0: Aplus0 += [i]
	  if i >= 0 and i %5 == 0 and i % 25 != 0: Aplus1 += [i]
	  if i <= 0 and i % 5 != 0: Aminus0 += [i]
	  if i <= 0 and i % 5 == 0 and i % 25 != 0: Aminus1 += [i]
	Aminus0, Aminus1, Aplus0, Aplus1 = sorted(Aminus0), sorted(Aminus1), sorted(Aplus0), sorted(Aplus1)
	answers = []
	if len(Aplus0) >= 3: answers += [Aplus0[-1] * Aplus0[-2] * Aplus0[-3]]
	if len(Aplus0) >= 2 and len(Aplus1) >= 1: answers += [Aplus1[-1] * Aplus0[-1] * Aplus0[-2]]
	if len(Aplus0) >= 1 and len(Aminus0) >= 2: answers += [Aplus0[-1] * Aminus0[0] * Aminus0[1]]
	if len(Aminus0) >= 2 and len(Aplus1) >= 1: answers += [Aplus1[-1] * Aminus0[0] * Aminus0[1]]
	if len(Aplus0) >= 1 and len(Aminus1) >= 1 and len(Aminus0) >= 1: answers += [Aminus1[0] * Aminus0[0] * Aplus0[-1]]
		25

	Идея состоит в том, что проверяемое условие это по сути то же самое, что и просто проверить само число на делимость на 25. Тогда можно перебрать несколько вариантов, с самыми большими числами, не делящимися на 5, с одним числом, делящимся на пять, а также не забыть про такие же варианты с отрицательными числами, но уже наибольшими по модулю.	

2 вариант

№	Правильный ответ	Балл
1.	Во второй строчке таблицы если ¬y ∧ x = 1, то x=1, y=0 и 1 → z = 1, то z = 1. Значит, оставшийся ноль может быть только y, следовательно, первая переменная y. Далее, если x ∧ ¬y = 0, то либо x=0, либо y=1, но в третьей строчке y = 0, значит, в оставшемся пустом месте для третьей переменной должен быть ноль, и вторая переменная z. Итого yzx. Теперь если вся функция G равна нуля, то каждое из F должно быть равно нулю. Из второй и 4 строчки получаем F(1, 1) = F(1, 0) = F(0, 1) = 0. Если бы F(0,0) был равен 0, то G всегда была бы 0, следовательно, F(0, 0) = 1 и F(x, y) = ¬x and ¬y	16
2.	1) От кучи 83 отнимаем три спички, получаем позицию (80, 80), далее ходим симметрично, побеждает первый игрок. Перед следующими задачами стоит убедиться (например, с помощью обычного перебора), что если игра начинается с одной кучи и в куче от 3 до 10 спичек, то проигрышные позиции только 6 и 10. 2) Разобьём игру на две параллельные, одна с кучей из 6 спичек, другая из 35. Легко заметить, например, с помощью дерева, что любой ход с кучей в 6 спичек приводит к моментальному выигрышу второго игрока, то есть это проигрышная позиция. Из кучи в 35 спичек можно отнять одну спичку и поделить на 2 кучи по 17. Получается, что Ада перевела Настю в позицию, где две параллельные игры проигрышные. Это тоже, очевидно, проигрышная позиция, потому что любой ход приводит к комбинации проигрышной и выигрышной позиции, а из такой комбинации можно перейти снова в позицию минус-минус. Если Настя походит в одной из куч по 13, Ада сделает симметричный ход. Если Настя походит в куче 6, то Ада сразу же закончит эту кучу. Побеждает Ада. 3) Если Ада как-то походит в куче 3, то любой её ход сразу же приведёт к позиции (8) (возможно, с двумя кучами по одной спичке, но они уже роли не играют). Тогда Настя ходит в (6, 1) и выигрывает (6 позиция проигрышная). Значит, у Ады есть ходы (5,3), (6, 1, 3), (5, 2, 3), (4, 3, 3). В (4, 3, 3) остаётся ровно 3 хода, значит, Настя походит последней в любом случае, в (6, 1, 3) можно любым способом убрать кучу из 3 спичек и получить позицию, эквивалентную позиции из 6 спичек (проигрышной), в (5, 3) или (5, 2, 3) можно разбить пятерку на 3 и 1 и получить позицию, эквивалентную (3, 4), в которой осталось ровно два хода и последний ход за Настей. Настя выигрывает. 4) Ада первым ходом ходит в позицию 9, 4 и выигрывает. Действительно, 9 выигрышная позиция (можно попасть в 6, проигрышную). Любой ход в куче 4 уничтожит эту кучу, что поставит Аду в выигрышную позицию. Если же Настя походит из 9 в 6 или (6,2), то получится комбинация выигрышной плюс	20

	проигрышной позиции для Ады. Если Настя походит в (7, 1, 4), Ада походит в (4, 2, 1, 4), где остаётся ровно два хода. Если Настя походит в (3, 5, 4), Ада походит в (3, 2, 2, 1, 4), где остаётся ровно два хода. Ада выигрывает.	
3.	1101192. Вычислять лучше всего с помощью Excel, выделив все случаи, когда робот может взять ровно 3 монеты и сымитировав их в таблице.	18

4.	M = int(input())
	FA = [[-1 for i in range(5)] for j in range(3*M)]
	def s(n):
	  k = 0
	  while n:
	  k += n%3
	  n //= 3
	  return k
	def f(n, m, op = 0):
	  if n>m: return 0
	  if n==m: return 1
	  ans = 0
	  if op != 1:
	  if FA[n+1][1] == -1:
	  FA[n+1][1] = f(n+1, m, 1)
	  ans += FA[n+1][1]
	  if op != 2:
	  if FA[n+2][2] == -1:
	  FA[n+2][2] = f(n+2, m, 2)
	  ans += FA[n+2][2]
	  if op != 3:
	  if FA[n+3][3] == -1:
	  FA[n+3][3] = f(n+3, m, 3)
	  ans += FA[n+3][3]
	  if op != 4:
	  if FA[n+s(n)][4] == -1:
	  FA[n+s(n)][4] = f(n+s(n), m, 4)
	  ans += FA[n+s(n)][4]
	  return ans
	print(f(1, M))
	Стоит обратить внимание, что обычное динамическое программирование будет работать очень долго на некоторых тестах. Стоит провести запись значений в специальный массив.
		21

5.	N = int(input())
	A = map(int, input().split())
	Aminus0, Aminus1, Aplus0, Aplus1 = [], [], [], []
	for i in A:
	  if i>=0 and i%2!=0: Aplus0 += [i]
	  if i>=0 and i%2==0 and i%4!=0: Aplus1 += [i]
	  if i<=0 and i%2!=0: Aminus0 += [i]
	  if i<=0 and i%2==0 and i%4!=0: Aminus1 += [i]
	Aminus0, Aminus1, Aplus0, Aplus1 = sorted(Aminus0), sorted(Aminus1), sorted(Aplus0), sorted(Aplus1)
	answers = []
	if len(Aplus0)>=3: answers += [Aplus0[-1] * Aplus0[-2] * Aplus0[-3]]
	if len(Aplus0)>=2 and len(Aplus1)>=1: answers += [Aplus1[-1] * Aplus0[-1] * Aplus0[-2]]
		25

	if len(Aplus0)>=1 and len(Aminus0)>=2: answers += [Aplus0[-1] * Aminus0[0] * Aminus0[1]] if len(Aminus0)>=2 and len(Aplus1)>=1: answers += [Aplus1[-1] * Aminus0[0] * Aminus0[1]] if len(Aplus0)>=1 and len(Aminus1)>=1 and len(Aminus0)>=1: answers += [Aminus1[0] * Aminus0[0] * Aplus0[-1]] Идея состоит в том, что проверяемое условие это по сути то же самое, что и просто проверить само число на делимость на 4. Тогда можно перебрать несколько вариантов, с самыми большими числами, не делящимися на 2, с одним числом, делящимся на 2, а также не забыть про такие же варианты с отрицательными числами, но уже наибольшими по модулю.	

3 вариант

№	Правильный ответ	Балл
1.	В первой и второй строчке таблицы x и z не должны меняться, значит, третья строчка y. Далее, если $z \wedge \neg x = 0$, то либо z=0, либо x=1, но в четвёртой строчке переменная 1 равна 1, а 2 равна 0, значит если сделать переменную 1 z, то это условие не выполнится. Итого xzy. Теперь если вся функция G равна нуля, то каждое из F должно быть равно нулю. Из второй строчки получаем $F(1, 1) = F(0, 1) = 0$. Если бы F(0,0) был равен 0, то G всегда была бы 0, следовательно, есть только 3 варианта для функции, из которых подходят два: $F(x, y) = \neg x$ and $\neg y$ и $F(x, y) = \neg y$	16
2.	1) Насте достаточно повторять симметрично ходы Ады для противоположной кучи, чтобы победить. Перед следующими задачами стоит убедиться (например, с помощью обычного перебора), что если игра начинается с одной кучи и в куче от 3 до 10 спичек, то проигрышные позиции только 6 и 10. 2) Разобьём игру на две параллельные, одна с кучей из 6 спичек, другая из 29. Легко заметить, например, с помощью дерева, что любой ход с кучей в 6 спичек приводит к моментальному выигрышу второго игрока, то есть это проигрышная позиция. Из кучи в 29 спичек можно отнять одну спичку и поделить на 2 кучи по 14. Получается, что Ада перевела Настю в позицию, где две параллельные игры проигрышные. Это тоже, очевидно, проигрышная позиция, потому что любой ход приводит к комбинации проигрышной и выигрышной позиции, а из такой комбинации можно перейти снова в позицию минус-минус. Если Настя походит в одной из куч по 13, Ада сделает симметричный ход. Если Настя походит в куче 6, то Ада сразу же закончит эту кучу. Побеждает Ада. 3) Если Ада как-то походит в куче 4, то любой её ход сразу же приведёт к позиции (8) (возможно, с двумя кучами по одной спичке, но они уже роли не играют). Тогда Настя ходит в (6, 1) и выигрывает (6 позиция проигрышная). Значит, у Ады есть ходы (5,4), (6, 1, 4), (5, 2, 4), (4, 3, 4). В (4, 3, 4) остаётся ровно 3 хода, значит, Настя походит последней в любом случае, в (6, 1, 4) можно любым способом убрать кучу из 3 спичек и получить позицию, эквивалентную позиции из 6 спичек (проигрышной), в (5, 4) или (5, 2, 4) можно разбить пятерку на 3 и 1 и получить позицию, эквивалентную (3, 4), в которой осталось ровно два хода и последний ход за Настей. Настя	20

	выигрывает. 4) Ада первым ходом ходит в позицию 8, 4 и выигрывает. Действительно, 8 выигрышная позиция (можно попасть в 6, проигрышную). Любой ход в куче 4 уничтожит эту кучу, что поставит Аду в выигрышную позицию. Если же Настя походит из 8 в 5 или (5,2), то получится комбинация выигрышной плюс проигрышной позиции для Ады. Если Настя походит в (6, 1, 4), Ада походит в (3, 2, 1, 4), где остаётся ровно два хода. Если Настя походит в (3, 4, 4), Ада походит в (3, 2, 1, 1, 4), где остаётся ровно два хода. Ада выигрывает.	
3.	356820. Вычислять лучше всего с помощью Excel, выделив все случаи, когда робот может взять ровно 3 монеты и сымитировав их в таблице.	18

4.	M = int(input())
	FA = [[-1 for i in range(5)] for j in range(3*M)]
	def s(n):
	  k = 0
	  while n:
	  k += n%7
	  n //= 7
	  return k
	def f(n, m, op = 0):
	  if n>m: return 0
	  if n==m: return 1
	  ans = 0
	  if op != 1:
	  if FA[n+1][1] == -1:
	  FA[n+1][1] = f(n+1, m, 1)
	  ans += FA[n+1][1]
	  if op != 2:
	  if FA[n+2][2] == -1:
	  FA[n+2][2] = f(n+2, m, 2)
	  ans += FA[n+2][2]
	  if op != 3:
	  if FA[n+3][3] == -1:
	  FA[n+3][3] = f(n+3, m, 3)
	  ans += FA[n+3][3]
	  if op != 4:
	  if FA[n+s(n)][4] == -1:
	  FA[n+s(n)][4] = f(n+s(n), m, 4)
	  ans += FA[n+s(n)][4]
	  return ans
	print(f(1, M))
	Стоит обратить внимание, что обычное динамическое программирование будет работать очень долго на некоторых тестах. Стоит провести запись значений в специальный массив.
		21

5.	N = int(input())
	A = map(int, input().split())
	Aminus0, Aminus1, Aplus0, Aplus1 = [], [], [], []
	for i in A:
	  if i>=0 and i%7!=0: Aplus0 += [i]
	  if i>=0 and i%7==0 and i%49!=0: Aplus1 += [i]
	  if i<=0 and i%7!=0: Aminus0 += [i]
	  if i<=0 and i%7==0 and i%49!=0: Aminus1 += [i]
	Aminus0, Aminus1, Aplus0, Aplus1 = sorted(Aminus0), sorted(Aminus1),
		25

	sorted(Aplus0), sorted(Aplus1) answers = [] if len(Aplus0)>=3: answers += [Aplus0[-1] * Aplus0[-2] * Aplus0[-3]] if len(Aplus0)>=2 and len(Aplus1)>=1: answers += [Aplus1[-1] * Aplus0[-1] * Aplus0[-2]] if len(Aplus0)>=1 and len(Aminus0)>=2: answers += [Aplus0[-1] * Aminus0[0] * Aminus0[1]] if len(Aminus0)>=2 and len(Aplus1)>=1: answers += [Aplus1[-1] * Aminus0[0] * Aminus0[1]] if len(Aplus0)>=1 and len(Aminus1)>=1 and len(Aminus0)>=1: answers += [Aminus1[0] * Aminus0[0] * Aplus0[-1]] Идея состоит в том, что проверяемое условие это по сути то же самое, что и просто проверить само число на делимость на 49. Тогда можно перебрать несколько вариантов, с самыми большими числами, не делящимися на 7, с одним числом, делящимся на 7, а также не забыть про такие же варианты с отрицательными числами, но уже наибольшими по модулю.	

4 Вариант

№	Правильный ответ	Балл
1.	Во второй строчке таблицы если x ∨ y = 0, то x=y=0. Значит, оставшаяся 1 может быть только z, следовательно, вторая переменная z. Далее, если ¬z ≡ y = 1, то либо y=0 и z=1, тогда x = 1, либо наоборот, но во второй строчке z = 1, значит, в оставшемся пустом месте для третьей переменной должен быть 1, и третья переменная x. Итого yzx. Теперь если вся функция G равна 1, то каждое из F должно быть равно 1. Из первой и третьей строчки получаем F(1, 1) = F(0, 0) = F(0, 1) = 1. Если бы F(1,0) был равен 1, то G всегда была бы 1, следовательно, F(1, 0) = 0 и F(x, y) = (¬x or y)	16
2.	1) От кучи 85 отнимаем две спички, получаем позицию (83, 83), далее ходим симметрично, побеждает первый игрок. Перед следующими задачами стоит убедиться (например, с помощью обычного перебора), что если игра начинается с одной кучи и в куче от 2 до 11 спичек, то проигрышные позиции только 4 и 10. 2) Разобьём игру на две параллельные, одна с кучей из 10 спичек, другая из 39. Легко заметить, например, с помощью дерева, что любой ход с кучей в 10 спичек приводит к выигрышу второго игрока, то есть это проигрышная позиция. Из кучи в 39 спичек можно отнять три спичку и поделить на 2 кучи по 18. Получается, что Ада перевела Настю в позицию, где две параллельные игры проигрышные. Это тоже, очевидно, проигрышная позиция, потому что любой ход приводит к комбинации проигрышной и выигрышной позиции, а из такой комбинации можно перейти снова в позицию минус-минус. Если Настя походит в одной из куч по 13, Ада сделает симметричный ход. Если Настя походит в куче 10, то Ада сразу же закончит эту кучу. Побеждает Ада. 3) Если Ада как-то походит в куче 3, то любой её ход сразу же приведёт к позиции (11) (возможно, с двумя кучами по одной спичке, но они уже роли не играют). Тогда Настя ходит в (4, 4) и выигрывает. Значит, у Ады есть ходы (9, 3), (7, 1, 3), (6, 2, 3), (5, 3, 3), (4, 4, 3). В (4, 4, 3) остаётся ровно 5 ходов, значит, Настя походит последней в любом случае, в (7, 1, 3) можно походить (3, 1,1,3)	20

	и там остаётся только 2 хода. В (9, 3) или (6, 2, 3) Настя должна походить (4, 2, 3), остаётся ровно 4 хода. В (5, 3, 3) надо походить (1, 1, 3, 3). Настя выигрывает. 4) Ада первым ходом ходит в позицию 5, 1, 6 и выигрывает. Действительно, из 5 есть либо ход (1, 1), который оставляет одну кучу с 6, а 6 — это выигрышная позиция (можно попасть в 4, проигрышную), либо ход (3), тогда Ада ходит (3, 2, 1) и остаётся ровно 2 хода. Остаются ходы (5, 4) (совмещение проигрышной и выигрышной позиции) либо (5, 2, 1), откуда надо идти в (3, 2, 1). Ада выигрывает.	
3.	395232. Вычислять лучше всего с помощью Excel, выделив все случаи, когда робот может взять ровно 3 монеты и сымитировав их в таблице.	18

4.	M = int(input())
	FA = [[-1 for i in range(5)] for j in range(3*M)]
	def s(n):
	  k = 0
	  while n:
	  k = n%10
	  n //= 10
	  return k
	def f(n, m, op = 0):
	  if n>m: return 0
	  if n==m: return 1
	  ans = 0
	  if op != 1:
	  if FA[n+1][1] == -1:
	  FA[n+1][1] = f(n+1, m, 1)
	  ans += FA[n+1][1]
	  if op != 2:
	  if FA[n+2][2] == -1:
	  FA[n+2][2] = f(n+2, m, 2)
	  ans += FA[n+2][2]
	  if op != 3 and n%10!=0:
	  if FA[n+3][3] == -1:
	  FA[n+3][3] = f(n+(n%10), m, 3)
	  ans += FA[n+3][3]
	  if op != 4:
	  if FA[n+s(n)][4] == -1:
	  FA[n+s(n)][4] = f(n+s(n), m, 4)
	  ans += FA[n+s(n)][4]
	  return ans
	print(f(1, M))Стоит обратить внимание, что обычное динамическое программирование будет работать очень долго на некоторых тестах. Стоит провести запись значений в специальный массив.
		21

5.	N = int(input())
	A = map(int, input().split())
	A0, A1, A2 = [], [], []
	for i in A:
	  if i%2!=0: A0 += [i]
	  if i%2==0 and i%4!=0: A1 += [i]
	  if i%4==0 and i%8!=0: A2 += [i]
	A0, A1, A2 = sorted(A0), sorted(A1), sorted(A2)
	answers = []
		25

	if len(A0)>=3: answers += [A0[-1] * A0[-2] * A0[-3]] if len(A0)>=2 and len(A1)>=1: answers += [A1[-1] * A0[-1] * A0[-2]] if len(A1)>=2 and len(A0)>=1: answers += [A0[-1] * A1[-1] * A1[-2]] if len(A0)>=2 and len(A2)>=1: answers += [A2[-1] * A0[-1] * A0[-2]] p r Идея состоит в том, что проверяемое условие это по сути то же самое, что и просто проверить само число на делимость на 8. Тогда можно перебрать несколько вариантов, с самыми большими числами, не делящимися на 2, с одним числом, делящимся на 4, с несколькими числами, делящимися на 2, выбрав самые большие.	

Информатика. 11 класс
Критерии оценивания

1. За правильный ответ на каждый вопрос с обоснованием 8 баллов, за ответ без обоснования либо ошибку в рассуждениях 4 балла.

2. За правильный обоснованный ответ на один вопрос ставятся пять баллов. Частичные баллы ставились за неполный перебор дерева, некоторые хорошие идеи, не приведшие к результату (например, о определении позиции в комбинированной позиции через позиции в каждой отдельной куче).

3. За небольшие ошибки (связанные либо с не везде правильно заполненными формулами, либо с неправильным переносом) ставилось 12-15 баллов. За правильно рассмотренный только один случай либо неучёт того, что кучах, где монета не берётся, надо писать ноль, ставилось 6 баллов.

4. За работающую слишком медленно программу 10 баллов. Если программа неправильная, но есть некоторые полезные идеи, то 5 баллов.

5. За работающую слишком медленно программу 10 баллов. Если не были учтены отрицательные числа (1-3 вариант) или была просто применена сортировка и поиск до первого большого числа, подходящего под условия 15 баллов. Если программа неправильная, но есть некоторые полезные идеи, то 5 баллов.