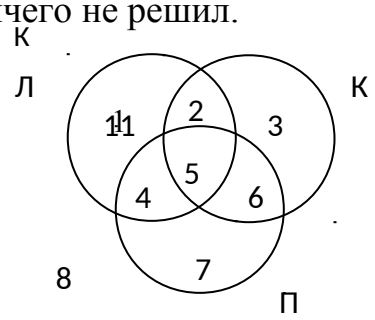
Информатика,10 класс

1 вариант

Решения и ответы

№	Ответ	Балл	Решение
1	2	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 что 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.  По условию задачи нам нужно найти область 5 –тех, кто решил все три задачи: на логику, комбинаторику и программирование. Количество школьников, соответствующих событиям каждой области **i**, будем обозначать через $\mathbf{N_i}$ Теперь посмотрим, что нам задано. 1) Всего в олимпиаде приняло участие 38 школьников. Т.е.

			$\mathbf{N_1 + N_2 + N_3 + N_4 + N_5 + N_6 + N_7 + N_8 = 38}$ 2) С задачей по теме «Логика» справились 16 человек. Это $\mathbf{N_1 + N_2 + N_4 + N_5 = 16}$ 3) Задачу по комбинаторике решили 17 участников. $\mathbf{N_2 + N_3 + N_5 + N_6 = 17}$ 4) Задачу по программированию решили 18 человек. $\mathbf{N_4 + N_5 + N_6 + N_7 = 18}$ 5) Трое участников не смогли решить ни одной задачи. $\mathbf{N_8 = 3}$ 6) 4 участника решили ровно две задачи: на логику и комбинаторику: $\mathbf{N_2 = 4}$ 7) Ещё 3 участника решили две задачи, но по логике и программированию: $\mathbf{N_4 = 3}$ 8) И 5 человек две задачи по комбинаторике и программированию: $\mathbf{N_6 = 5}$ 9) Количество школьников, не решивших ни задачу по программированию, ни по комбинаторике, равно 10) Это все области, лежащие за пределами кругов К и П, т.е. области 1 и 8. $\mathbf{N_1 + N_8 = 10}$ 11) По условию задачи $\mathbf{N_8 = 3.}$ Отсюда, $\mathbf{N_1 = 10 - N_8 = 10 – 3 = 7}$ 12) В круге «Логика» находится 16 человек. Это $\mathbf{N_1 + N_2 + N_4 + N_5 = 16.}$ Мы уже знаем, что $\mathbf{N_2 = 4,\ N_4 = 3,}$ **N1** – мы нашли в предыдущем пункте. Значит, $\mathbf{N_5 = 16 - N_1 - N_2 - N_4 = 16 - 7 - 4 – 3 = 2.}$ **Ответ: 2** участника решили все три задачи
2	2520	15	Сколько различных слов можно получить, переставляя буквы в слове задание? **Первый способ решения**. Для решения этой задачи нужно найти количество перестановок с повторами. Оно находится по следующей формуле: $Pn(k1,k2,..,kn)=n!/K1!*k2!*..kn!,$ где $k_1+k_2+..+k_n=n$, а k_i – количество повторов каждого элемента (в нашем случае – количество повторов каждой буквы в слове). В слове «задание» 7 букв, из них две буквы «а», все остальные буквы встречаются по 1 разу. Число

			повторений: буква «з» – k_1=1, буква «а» – k_2=2, буква «д» – k_3=1, буква «н» – k_4=1, буква «и» – k_5=1, буква «е» – k_6=1. **P6(1,2,1,1,1,1)=7!/1!2! 1! 1! 1! 1! 1!=** 5040:2 = 2520. **Второй способ**. В этом слове две буквы «а», а все остальные буквы разные. Временно будем считать разными и буквы А, обозначив их через A_1 и A_2. Посчитаем при этом предположении количество перестановок. Оно будет равно n!= 7!= 5040 разных слов. Однако те слова, которые получаются друг из друга перестановкой букв A_1 и A_2, на самом деле одинаковы. Таким образом, полученные 5040 слов разбиваются на пары одинаковых слов. Поэтому разных слов всего 5040:2 = 2520.
3	304	10	Из условия задачи следует, что нам нужно найти сумму чисел, хранящихся в ячейках заданной области: ячейках, отстоящих от места высадки не далее, чем на 9 шагов от места высадки, которое находится в ячейке L1. Проблема заключается в том, что пират не может передвигаться по диагонали, а только по горизонтали или вертикали. Поэтому в пределах той же строки 1 он может пойти на от L1 влево на 9 – C1, от L1 вправо на 9 – U1. Если же он спустится в строку 2, то он уже потратит один шаг на спуск, а, значит, сможет сдвинуться от L2 влево на 8 – D2, от L2 вправо на 8 – D2. В строке 3 диапазон подсчёта уже сузится до 6 (E3:S3) и т.д. В строке 10 он может попасть только в ячейку L10. Обратите также внимание, что в условиях не было ограничения на количество ходок пирата. Он выходит, собирает, возвращается и снова движется из точки высадки. Итак, для решения задачи нужно просуммировать значения в нужном диапазоне в каждой из строк 1-10. Для суммирования значений необходимо воспользоваться формулой **=СУММ(СУММ(C1:U1);**

			СУММ(D2:T2);СУММ(E3:S3);СУММ(F4:R4);СУММ(G5:Q5);СУММ(H6:P6);СУММ(I7:O7);СУММ(J8:N8);СУММ(K9:M9);СУММ(L10)) в любой ячейке, лежащей вне диапазона A1:U40. В результате вычислений сумма будет равна **304.**

4	Написанная программа	25	Для решения задачи можно использовать массивы, но так как нам не требуется выполнять действия над всеми членами последовательности одновременно, наиболее оптимальным будет в каждый момент времени хранить только 4 текущие члена последовательности. И еще одна переменная нужна для вычисления нового члена последовательности.
			Алгоритм решения задачи следующий:
			1. Считываем n – номер члена последовательности.
			2. Считываем первые четыре члена последовательности.
			В переменной x будем хранить новый член последовательности.
			По заданию нам нужно вычислить члены последовательности с 5 и до n-ного (поскольку каждый из них вычисляется на основе предыдущих четырех). Поэтому выполняем цикл, счетчик которого меняется от 5 до n.
			3. В цикле повторяем следующее:
			3.1) Суммируем текущие четыре члена последовательности.
			3.2) Находим последнюю цифру от этой суммы как остаток от деления на 10, запоминаем в переменную x.
			3.3) Выполняем сдвиг по последовательности: число, которое на текущем шаге было вторым, становится первым для следующего шага. Число, которое было третьим на данном этапе, будет вторым для следующего этапа. Третьим станет текущий «четвертый» член последовательности. А последним – четвёртым станет новый член последовательности, записанный в x. Таким образом, мы определим четыре значения для следующего члена последовательности.
			3.4) Повторяем действия цикла.
			После выхода из цикла в переменной x будет хранится *n-ый* член последовательности.
			
			Пример программы на языке C++:
			`#include <iostream>`
			`using namespace std;`
			
			`int main()`

			{
			int a,b,c,d,n,x,i=5;
			cin>>n;
			if(n>4&&n<1000)
			{
			cin>>a>>b>>c>>d;
			x=0;
			while(i<=n)
			{
			x=(a+b+c+d)%10;
			a=b;
			b=c;
			c=d;
			d=x;
			i++;
			}
			cout<<x;
			}
			return 0;
			}

5.	Написанная программа	30	Выражение «найти максимальное значение среди светофоров, установленных на каждом из перекрестков», означает, что для каждого перекрестка нужно найти число перекрестков, с которыми он связан дорогами, а уже затем – найти среди них максимальное значение. Для решения задачи можно использовать граф, показывающий связь между перекрестками, который может быть представлен, например, матрицей смежности. Но наиболее оптимальным способом хранения данных в данном случае будет представление с помощью массива, в котором для каждого перекрестка будет храниться количество светофоров (т.е. связей с другими). На вход программе подается количество перекрестков N и количество тоннелей M. Далее идут M строк, в каждой по три числа через пробел – *номер_тоннеля перекресток_i перекресток_j*. По условию задачи дорога двусторонняя, т. е. строка 1 1 2 означает, что тоннель № 1 связывает перекрестки 1 и 2. Номер тоннеля для нашей задачи никакой роли не играет. Важно, какие номера перекрестков встречаются в каждой строке. Алгоритм будет иметь следующий вид: 1. считываем количество перекрестков N и количество тоннелей (строк) M.

2. объявляем целочисленный массив mas размером N для хранения количества связей для каждого перекрестка. Т.е. каждый элемент этого массива – это счётчик для соответствующего перекрестка.
3. Повторяем в цикле, который будет работать M раз, следующие действия:
3.1) считываем строку с информацией о тоннеле вида: num i j
Номер тоннеля num нас не интересует. i и j –это номера перекрестков.
3.2) увеличиваем счётчик встречаний для тоннелей i и j, т.е. прибавляем 1 к mas[i] и mas[j] .
3.3) повторяем действия.
После этого цикла в массиве mas хранится количество светофоров (т.е. связей с другими).
4. Теперь остается пройти по массиву mas и найти максимальное значение.

Пример программы на языке C++

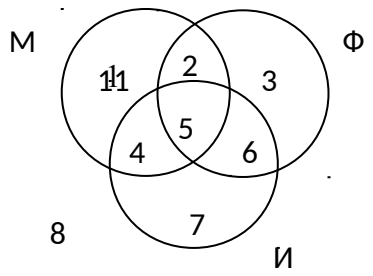
```
#include <iostream>
using namespace std;


int main()
{
    int N,M, mas[1000];
    int num, i, j, k;

    cin >> N;
    cin >> M;
    if (M > 0)
    {
      for (int i = 0; i < N; i++)
          mas[i] = 0;

      for (int k = 1; k <= M; k++)
      {
          cin >> num >> i >> j;
          mas[i-1]++;
          mas[j-1]++;
      }


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

			if (mas[i]>max) max=mas[i];
			}
			cout << max<<endl;
			}
			return 0;
			}

Информатика,10 класс

2 вариант

Решения и ответы

№	Ответ	Балл	Решение
1	2	20	Для решения задачи необходимо построить диаграмму Эйлера-Венна. На ней отобразим три множества событий. Множество М – множество школьников, которые посещают дополнительные кружки по математике, множество Ф – по физике и множество П – по информатике. На диаграмме обозначим каждую область цифрой: так, например, область что 5 – это **М ∩Ф ∩ И** – те, кто посещает все три кружка: по математике, физике и информатике. В других обозначениях можно записать, что 5 – это М **И** Ф **И** И или М **&**Ф **&** И. Область 2 – это **(М ∩ Ф) ∩⌐ И** или (**М &Ф**) **&** ⌐**И** – те, кто посещает кружок по математике и кружок по физике, но не посещает кружок по информатике. Область 1 – (**М ∩ ⌐ Ф)** **∩⌐ И** или (М **&⌐ Ф**) **&** ⌐И – те, кто ходит только в кружок по математике, но не на физику и информатику. И так далее. Последняя область 8 – те, кто не ходят ни в один из кружков.  По условию задачи нам нужно найти область 5 –тех, кто посещает все три кружка: по математике, физике и информатике. Количество школьников, соответствующих событиям каждой области **i**, будем обозначать через $\mathbf{N_i}$ Теперь посмотрим, что нам задано. 1) Всего в классе учится 36 школьников. Т.е. $\mathbf{N_1 + N_2 + N_3 + N_4 + N_5 + N_6 + N_7 + N_8 = 36}$ 2) Кружок по математике посещают 18 человек. Это $\mathbf{N_1 + N_2 + N_4 + N_5 = 18}$

			3) В кружок по физике ходят 14 участников. $\mathbf{N_2 + N_3 + N_5 + N_6 = 14}$ 4) В кружок по информатике ходят 12 человек. $\mathbf{N_4 + N_5 + N_6 + N_7 = 12}$ 5) Шесть школьников не занимаются ни в одном из кружков: $\mathbf{N_8 = 6}$ 6) 8 школьников посещают одновременно математический и физический кружок: $\mathbf{N_2 + N_5 = 8}$ 7) 5 школьников посещают кружки по математике и информатике: $\mathbf{N_4 + N_5 = 5}$ 8) 3 человека – кружки по физике и информатике: $\mathbf{N_6 + N_5 = 3}$ 9) Количество школьников, не занимающихся ни физикой, ни информатикой, равно 13. Это все области, лежащие за пределами кругов Ф и И, т.е. области 1 и 8. $\mathbf{N_1 + N_8 = 13.}$ По условию задачи $\mathbf{N_8 = 6.}$ Отсюда, $\mathbf{N_1 = 13 - N_8 = 13 - 6 = 7}$ 10) В круге «Математика» находится 18 человек. Это $\mathbf{N_1 + N_2 + N_4 + N_5 = 18.}$ Мы уже знаем, что $\mathbf{N_1 = 7.}$ Значит - $\mathbf{N_2 + N_4 + N_5 = 18-7 = 11.}$ Из пунктов 6 и 7 мы знаем, что $\mathbf{N_2 + N_5 = 8}$, а $\mathbf{N_4 + N_5 = 5}$. Сложив эти два выражения, получим $\mathbf{N_2 + N_4 + 2*N_5 = 13.}$ Получили систему: $\mathbf{N_2 + N_4 + 2*N_5 = 13}$ $\mathbf{N_2 + N_4 + N_5 = 11}$ Отнимем от первого уравнения второе и получим $\mathbf{N_5 = 2}$ **Ответ: 2** школьника посещают все три кружка.
2		15	Сколько различных слов можно получить, переставляя буквы в слове решение? **Первый способ решения**. Для решения этой задачи нужно найти количество перестановок с повторами. Оно находится по следующей формуле: $Pn(k1,k2,..,kn)=n!/K1!*k2!*..kn!,$ где $k_1+k_2+..+k_n=n$, а k_i – количество повторов каждого элемента (в нашем случае – количество повторов каждой буквы в слове). В слове «решение» 7 букв, из них три буквы «е», все остальные буквы встречаются по 1 разу. Число

			повторений: буква «р» – k_1=1, буква «е» – k_2=3, буква «ш» – k_3=1, буква «н» – k_4=1, буква «и» – k_5=1. **P6(1,3,1,1,1)=7!/1!3! 1! 1! 1!=** 5040:6 = 840. **Второй способ**. Три места для буквы Е можно выбрать C^3_7 =n!/k!(n-k)! способами.=7!/3!*4! Остальные 4 буквы можно переставлять по 4 оставшимся местам 4! способами. Итого 7!:3! = 5040:6 = 840 слов.
3	486	10	Из условия задачи следует, что нам нужно найти сумму чисел, хранящихся в ячейках заданной области: ячейках, отстоящих от поселения не далее, чем на 8 клеток. Место поселения в файле находится в ячейке J20. При этом еще задается условие на ячейки для суммирования: мы можем суммировать только значения <=12. Проблема заключается в том, что поселенец не может передвигаться по диагонали, а только по горизонтали или вертикали. Поэтому в пределах той же строки 20 он может пойти на от J20 влево на 8 – до B20, а вправо – до R20. Если же он спустится в строку 21, то он уже потратит один шаг на спуск, а, значит, сможет сдвинуться от J20 влево на 7 – C21, а вправо на 7 – Q21. В строке 22 диапазон подсчёта уже сузится до 6 ячеек (**D22:P22**) и т.д. В строке 28 он может попасть только в ячейку J28. С движением вверх наблюдается та же ситуация. Обратите также внимание, что в условиях не было ограничения на количество ходок поселенца. Он выходит, собирает, возвращается и снова движется из поселения. Итак, для решения задачи нужно просуммировать только значения <=12 в нужном диапазоне в каждой из строк 12-28. Для суммирования значений необходимо воспользоваться формулой: **=СУММ(СУММЕСЛИ(J12;"<=12");СУММЕСЛИ(I13:K13;"<=12");СУММЕСЛИ(H14:L14;"<=12");СУММЕСЛИ(G15:M15;"<=12");СУММЕСЛИ(F16:N16;"<=12");СУММЕСЛИ(E17:O17;"<=12");СУММЕСЛИ(D18:P18;"<=12");СУММЕСЛИ(C19:Q19;"<=12")**

			;СУММЕСЛИ(B20:R20;"<=12");СУММЕСЛИ(C21:Q21;"<=12");СУММЕСЛИ(D22:P22;"<=12");СУММЕСЛИ(E23:O23;"<=12");СУММЕСЛИ(F24:N24;"<=12");СУММЕСЛИ(G25:M25;"<=12");СУММЕСЛИ(H26:L26;"<=12");СУММЕСЛИ(I27:K27;"<=12");СУММЕСЛИ(J28;"<=12")) в любой ячейке, лежащей вне диапазона A1:W40. В результате вычислений сумма будет равна **486.**
4	Написанная программа	25	Для решения задачи можно использовать массивы, но так как нам не требуется выполнять действия над всеми членами последовательности одновременно, наиболее оптимальным будет в каждый момент времени хранить только 3 текущие члена последовательности. И еще одна переменная нужна для вычисления нового члена последовательности. Алгоритм решения задачи следующий: 1. Считываем n – номер члена последовательности. 2. Считываем первые три члена последовательности. В переменной x будем хранить новый член последовательности. По заданию нам нужно вычислить члены последовательности с 4 и до n-ного (поскольку каждый из них вычисляется на основе предыдущих четырех). Поэтому выполняем цикл, счетчик которого меняется от 4 до n. 3. В цикле повторяем следующее: 3.1) Суммируем текущие три члена последовательности. 3.2) Суммируем цифры полученного числа. 3.3) Перебираем его цифры так: в цикле повторяем следующее: берем последнюю цифру как остаток от деления на 10, добавляем к сумме, делим число на 10 и повторяем цикл, пока число>0. Так мы получаем новый член последовательности. 3.4) Выполняем сдвиг по последовательности: число, которое на текущем шаге было вторым, становится первым для следующего шага. Число, которое было третьим на данном этапе, будет вторым для следующего этапа. Третьим станет новый член последовательности, записанный в x. Таким образом, мы определим три значения для следующего члена последовательности. 3.5) Повторяем действия.

			После выхода из цикла в переменной x будет хранится *n-ый* член последовательности.
			4. Выводим его значение.
			
			Пример программы на языке C++:
			`#include <iostream>`
			`using namespace std;`
			
			`int main()`
			`{`
			`  int a,b,c,d,n,x,i=5;`
			`  cin>>n;`
			`  if(n>4&&n<1000)`
			`  {`
			`  cin>>a>>b>>c>>d;`
			`  x=0;`
			`  while(i<=n)`
			`  {`
			`  x=(a+b+c+d)%10;`
			`  a=b;`
			`  b=c;`
			`  c=d;`
			`  d=x;`
			`  i++;`
			`  }`
			`  cout<<x;`
			`  }`
			`  return 0;`
			`}`

5	Написанная программа	30	Для решения задачи можно использовать матрицу, в каждой i-ой строке которой будут храниться результаты встреч i-ой команды со всеми остальными командами. Но наиболее оптимальным способом хранения данных в данном случае будет представление с помощью массива, в котором для каждого элемента будет храниться количество набранных очков, которые мы будем сразу вычислять при считывании данных. Алгоритм будет иметь следующий вид: 1. Считываем количество команд N. 2. Объявляем целочисленный массив mas размером N для хранения количества очков (можно объявить массив из максимально допустимого значения 1000, а реально использовать N из них). 3. Повторяем в цикле, который будет работать N*(N-1)/2 раз – число возможных пар из N элементов, следующие действия:

3.1) считываем строку с информацией об игре вида *номер_команды1 номер_команды2 счёт*, например так: x y c1 c2. Здесь x – номер первой команды, y – второй. c1 – результат первой команды, c2 – второй.
3.2) Сравниваем c1 и c2. Если c1>c2, то победила команда x. Значит нужно увеличить количество ее очков на 3, т.е. прибавить 3 к mas[x]. Если c1<c2, то победила команда y. Увеличиваем количество ее очков на 3, т.е. прибавляем 3 к mas[y]. Если c1=c2, то ничья. Увеличиваем на 1 оба счётчика: mas[x] и mas[y].
3.3) повторяем действия.
После этого цикла в массиве mas хранится количество очков, заработанных каждой командой.
4. Теперь остается пройти по массиву mas, найти максимальное значение и номер элемента, в котором оно хранится.
5. Выводим на экран найденные значения.

Пример программы на языке C++

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

int main()
{
    int N,M, mas[1000];
    int num, i, x, y, c1, c2, max=0;

    cin >> N;
    M=N*(N-1)/2;
    if (M > 0)
    {
      for (int i = 0; i < N; i++)
          mas[i] = 0;

      for (int i = 1; i <= M; i++)
      {
          cin >> x >> y>>c1>>c2;
          if(c1>c2) mas[x-1]=mas[x-1]+3;
            if(c1<c2) mas[y-1]=mas[y-1]+3;
          if(c1==c2) {
                mas[x-1]++;
                 mas[y-1]++;
          }
      }
```

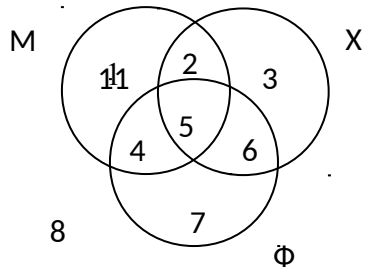
```
      num=0;
      for (int i = 0; i < N; i++)
      {
            if (mas[i]>max) {
                  max=mas[i];
                  num=i;
            }
      }
      cout <<num+1<< " " << max<<endl;
      }
     return 0;
}
```


Информатика,10 класс

3 вариант

Решения и ответы

№	Ответ	Балл	Решение
1.	17	20	Для решения задачи необходимо построить

			диаграмму Эйлера-Венна. На ней отобразим три множества событий. Множество М – множество школьников, которые посещают дополнительные кружки по математике, множество Х – по химии и множество Ф – по физике. На диаграмме обозначим каждую область цифрой: так, например, область что 5 – это **М ∩Х ∩ Ф** – те, кто посещает все три кружка: по математике, химии и физике. В других обозначениях можно записать, что 5 – это М **И** Х **И** Ф или М **&Х &** Ф. Область 2 – это **(М ∩ Х) ∩¬Ф** или (**М &Х**) **&** ¬**Ф** – те, кто посещает кружок по математике и кружок по химии, но не посещает кружок по физике. Область 1 – (**М ∩ ¬ Х) ∩¬ Ф** или (М **&¬ Х**) **&** ¬**Ф** – те, кто ходит только в кружок по математике, но не на химию и физику. И так далее. Последняя область 8 – те, кто не ходят ни в один из кружков.  По условию задачи нам нужно найти количество школьников, которые посещают только по одному кружку. В наших обозначениях это будет объединение областей 1, 3 и 7. Количество школьников, соответствующих событиям каждой области **i**, будем обозначать через $\mathbf{N_i}$ Теперь посмотрим, что нам задано. 1) Всего в классе учится 32 школьника. Т.е. $\mathbf{N_1 + N_2 + N_3 + N_4 + N_5 + N_6 + N_7 + N_8 = 32}$ 2) Кружок по математике посещают 17 человек. Это $\mathbf{N_1 + N_2 + N_4 + N_5 = 17}$ 3) В кружок по химии ходят 10 участников. $\mathbf{N_2 + N_3 + N_5 + N_6 = 10}$ 4) В кружок по физике ходят 13 человек. $\mathbf{N_4 + N_5 +}$

			$N_6 + N_7 = 13$ 5) Пять школьников не занимаются ни в одном из кружков: $N_8 = 5$ 6) 5 школьников посещают одновременно математический и химический кружок: $N_2 + N_5 = 5$ 7) 7 школьников посещают кружки по математике и физике: $N_4 + N_5 = 7$ 8) 4 человека – кружки по химии и физике: $N_5 + N_6 = 4$ 9) Количество школьников, не занимающихся ни в кружке по математике, ни в кружке по физике, равно 9. Это все области, лежащие за пределами кругов М и Ф, т.е. области 3 и 8. $N_3 + N_8 = 9$. По условию задачи $N_8 = 5$. Отсюда, $N_3 = 9 - N_8 = 9 - 5 = 4$ 10) В круге «Химия» находится 10 человек. Это $N_2 + N_3 + N_5 + N_6 = 10$. Мы уже знаем, что $N_3 = 4$. Значит - $N_2 + N_5 + N_6 = 10-4=6$. Из пунктов 6 и 8 мы знаем, что $N_2 + N_5 = 5$, а $N_5 + N_6 = 4$. Сложив эти два выражения, получим $N_2 + 2*N_5 + N_6 = 9$. Получили систему: $N_2 + 2*N_5 + N_6 = 9$ $N_2 + N_5 + N_6 = 6$ Отнимем от первого уравнения второе и получим $N_5 = 3$ 11) Так как $N_2 + N_5 = 5$, а $N_5 = 3$, отсюда $N_2 = 2$. 12) Из пункта $N_4 + N_5 = 7$, а $N_5 = 3$, отсюда $N_4 = 4$. 13) В круге «Математика» находится 17 человек. Это $N_1 + N_2 + N_4 + N_5 = 17$. Мы уже нашли $N_2 = 2$, $N_4 = 4$, $N_5 = 3$. Следовательно, $N_1 = 17-2-4-3=8$ 14) Из пунктов 8 мы знаем, что $N_5 + N_6 = 4$. Так как $N_5 = 3$, отсюда $N_6 = 1$. 15) В кружок по физике ходят 13 человек. $N_4 + N_5 + N_6 + N_7 = 13$. Мы уже нашли $N_4 = 4$, $N_5 = 3$, $N_6 = 1$. Следовательно, $N_7 = 13-4-3-1=5$. 16) По условию задачи нам нужно найти количество школьников, которые посещают только по одному кружку. В наших обозначениях это будет объединение областей 1, 3 и 7. Т.е. нужно найти $N_1 + N_3 + N_7$. Они уже нам известны.

			Получаем: $\mathbf{N_1+ N_3 + N_7=8+4+5=17.}$ **Ответ: 17** школьников посещают только по одному кружку.
2.	2520	15	Сколько различных слов можно получить, переставляя буквы в слове вариант? **Первый способ решения**. Для решения этой задачи нужно найти количество перестановок с повторами. Оно находится по следующей формуле: Pn(k1,k2,..,kn)=n!/K1!*k2!*..kn!, где k1+k2+..+kn=n, а ki – количество повторов каждого элемента (в нашем случае – количество повторов каждой буквы в слове). В слове «вариант» 7 букв, из них две буквы «а», все остальные буквы встречаются по 1 разу. Число повторений: буква «в» – k_1=1, буква «а» – k_2=2, буква «р» – k_3=1, буква «и» – k_4=1, буква «н» – k_5=1, буква «т» – k_6=1. **P6(1,2,1,1,1,1)=7!/1!2! 1! 1! 1! 1! 1!=** 5040 : 2 = 2520. **Второй способ**. В этом слове две буквы «А», а все остальные буквы разные. Временно будем считать разными и буквы А, обозначив их через $А_1$ и $А_2$. Посчитаем при этом предположении количество перестановок. Оно будет равно n!= 7! = 5040 разных слов. Однако те слова, которые получаются друг из друга перестановкой букв $А_1$ и $А_2$, на самом деле одинаковы. Таким образом, полученные 5040 слов разбиваются на пары одинаковых слов. Поэтому разных слов всего 5040:2 = 2520.
3.	736	10	Из условия задачи следует, что нам нужно найти сумму чисел, хранящихся в ячейках заданной области: ячейках, отстоящих от поселения не далее

			чем на 10 строк и 10 столбцов в обе стороны. При этом еще задается условие на ячейки для суммирования: мы можем суммировать только значения >0. Место поселения в файле находится в ячейке K35. Определим диапазон подсчёта: от K35 влево на 10 и вверх на 10 – A25, от K35 вправо на 10 (вниз не нужно, так как ниже этой строки данных нет) – U35. Для суммирования только значений >0 необходимо воспользоваться формулой **==СУММЕСЛИ(A25:U35;">0")** в любой ячейке, лежащей вне диапазона A35:U35. В результате вычислений сумма будет равна **736.**
4.	Написанная программа	25	Для решения задачи можно использовать массивы, но так как нам не требуется выполнять действия над всеми членами последовательности одновременно, наиболее оптимальным будет в каждый момент времени хранить только 3 текущие члена последовательности. И еще одна переменная нужна для вычисления нового члена последовательности. Алгоритм решения задачи следующий: 1. Считываем n – номер члена последовательности. 2. Считываем первые три члена последовательности. В переменной x будем хранить новый член последовательности. В переменной `max будем` хранить максимальную цифру в получаемом числе. По заданию нам нужно вычислить члены последовательности с 4 и до n-ного (поскольку каждый из них вычисляется на основе предыдущих четырех). Поэтому выполняем цикл, счетчик которого меняется от 4 до n. В цикле повторяем следующее: 3. суммируем текущие три члена последовательности, затем ищем максимальную цифру полученного числа. 4. В переменную max присваиваем 0. 5. Перебираем цифры полученного результата суммы в цикле следующим образом:

			5.1) берем последнюю цифру как остаток от деления на 10,
			5.2) сравниваем с текущей хранимой максимальной цифрой `max`, если найденная цифра больше, то заносим ее в `max`.
			5.3) делим число на 10 и повторяем цикл, пока число>0.
			Полученное значение переменной max будет являться самой большой цифрой, встречавшейся в считанных числах. Так мы получаем новый член последовательности x.
			6. Выполняем сдвиг по последовательности: число, которое на текущем шаге было вторым, становится первым для следующего шага. Число, которое было третьим на данном этапе, будет вторым для следующего этапа. Третьим станет новый член последовательности, записанный в x. Таким образом, мы определим три значения для следующего члена последовательности.
			7. Повторяем действия.
			После выхода из цикла, в значении переменной x будет хранится *n-ый* член последовательности.
			8. Выводим его на экран.
			
			Пример программы на языке C++:
			`#include <iostream>`
			`using namespace std;`
			
			`int main()`
			`{`
			`  int a,b,c,n,x,k,max=0,i=4;`
			
			`  cin>>n;`
			`  if(n>3&&n<1000)`
			`  {`
			`  cin>>a>>b>>c;`
			`  x=0;`
			`  while(i<=n)`
			`  {`
			`  x=a+b+c;`
			`  k=0;`
			`  max=0;`
			`  while(x>0)`
			`  {`

			k=x%10;
			if(k>max) max=k;
			x=x/10;
			}
			x=max;
			a=b;
			b=c;
			c=x;
			i++;
			}
			cout<<x;
			}
			return 0;
			}

5.	Написанная программа	30	Для решения задачи можно использовать матрицу отношений, в которой для каждой i-ой строки (т.е. i-го школьника) будут храниться результаты бросков (попаданий/промахов) в каждого j-го школьника. Но наиболее оптимальным способом хранения данных в данном случае будет представление с помощью массива, в котором для каждого элемента будет храниться количество попаданий, которые мы будем сразу вычислять при считывании данных. Также обратите внимание, что при вводе мы считываем информацию о бросках с попаданием, но по условию задания нам нужно найти номер школьника, который промахнулся наибольшее число раз. Очевидно, что это означает, что нам нужно найти школьника с наименьшим количеством попаданий. Из этого следует, что нам нужно пройти по массиву, хранящему количество попаданий для каждого школьника, и найти номер элемента с минимальным значением. Алгоритм будет иметь следующий вид: 1. Считываем количество школьников N. 2. Объявляем целочисленный массив mas размером N для хранения количества попаданий (можно объявить массив из максимально допустимого значения 1000, а реально использовать N из них). 3. Считываем количество бросков с попаданием M. 4. Повторяем в цикле, который будет работать M раз, следующие действия:

4.1) считываем строку с информацией о броске с попаданием вида *номер_школьника1 номер_школьника2*, например так: x y. Здесь x – номер школьника, который попал в школьника y.

4.2) На самом деле в данном случае нам неважна информация о том, в кого попал школьник, важно просто понимать, что он в кого-то попал, а значит можно увеличить счётчик его попаданий, т.е. увеличить на 1 значение mas[x].

4.3) повторяем действия.

После этого цикла в массиве mas хранится количество попаданий для каждого школьника.

5. Теперь остается пройти по массиву mas, найти минимальное значение и номер элемента, в котором оно хранится.

6. Выводим на экран найденный номер.

Пример программы на языке C++

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


int main()
{
    int N,M, mas[1000];
    int num, i, x, y, min;

    cin >> N;
    cin >> M;
    if (M > 0)
    {
      for (int i = 0; i < N; i++)
          mas[i] = 0;

      for (int i = 1; i <= M; i++)
      {
          cin>> x >> y;
          mas[x-1]++;

          }


     int min=N;
     num=0;
     for (int i = 0; i < N; i++)
     {
          if (mas[i]<min) {
```

			min=mas[i];
			num=i;
			}
			}
			cout <<num+1<<endl;
			}
			return 0;
			}