Задача №4.

На каждой клетке игрового поля размером $M \times N$ клеток можно поместить либо дозорную башню с лучниками, либо стену, либо оставить его пустым. Лучники на башне, находящейся по координатам (x, y), могут атаковать другие башни по координатам (x_{-2}, y_{-2}), (x_{-2}, y_2), (x_2, y_{-2}) и (x_2, y_2), в том случае, когда они не разделены стеной.

На вход подается карта, на который отмечены свободные поля и поля с размещенными на них объектами. Ваша задача – разместить на карте как можно больше дополнительных башен, лучники на которых ***не будут*** атаковать друг друга.

Входные данные:

На вход может быть подано несколько тестов, каждый из которых имеет следующую структуру:

- Первая строка содержит два положительных целых числа M и N, разделенных пробелом
- Следующие M строк содержат N символов без пробелов, описывающих игровое поле, где
  - F – свободное поле
  - G – поле с башней
  - P – поле со стеной

Ввод данных считается законченным, когда M и N равны 0.

Выходные данные:

Целые числа, соответствующие количеству дополнительных башен.

Пример:

Ввод:	***Вывод:***
5 3 FPF FFF FGG FFF FPF 0 0	3 6

****

Требования к оформлению задач по программированию:

1) *Программы должны быть написаны на одном из языков: C, C++, Pascal*
2) *Полностью оформленная задача должна содержать:*
   - *программу, выполняющую необходимые операции для всех допустимых данных;*
   - *операции с файлами входных и выходных данных **или** понятный пользователю интерфейс ввода исходных данных;*
   - *комментарии к тексту программы, облегчающие ее понимание.*

Невыполнение вышеуказанных требований влечет за собой снижение получаемых за задачи баллов

РБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

² 252

1	2	3	4	Σ
6	6	5	10	27

заполняется жюри!

54

ПИСЬМЕННАЯ РАБОТА УЧАСТНИКА ОЛИМПИАДЫ ШКОЛЬНИКОВ СПбГУ 2016–2017

заключительный этап

Предмет (комплекс предметов) Олимпиады ИНФОРМАТИКА (10-11 КЛАССЫ)

Город, в котором проводится Олимпиада Казань

Дата 20.03.17

****

Вариант 10

****

Задача №1.

На планете «Лютые лютики» имеется свой набор символов для использования в именах жителей. Алфавит состоит из $F > 10$ символов. Имена всех жителей планеты состоят из $D > 4$ букв. Правитель планеты требует, чтобы кто-нибудь смог создать таблицу всех возможных имен, при учете, что никакая буква в имени не повторяется более или равно 4 раза.

Входные данные: Размер алфавита, Алфавит строкой, длина имени

Выходные данные: строки имен (в файл или на экран).

****

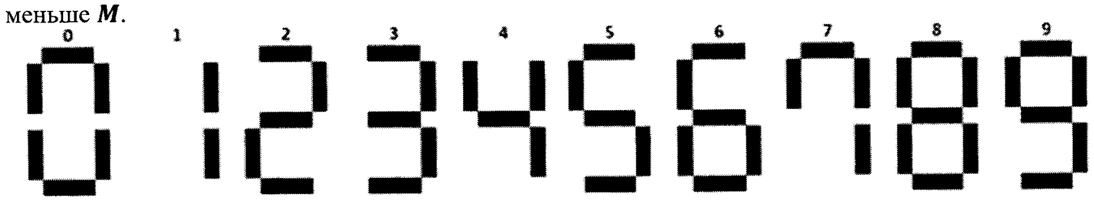
Требования к оформлению задач по программированию:

3) *Программы должны быть написаны на одном из языков: C, C++, Pascal*
4) *Полностью оформленная задача должна содержать:*
   - *программу, выполняющую необходимые операции для всех допустимых данных;*
   - *операции с файлами входных и выходных данных **или** понятный пользователю интерфейс ввода исходных данных;*
   - *комментарии к тексту программы, облегчающие ее понимание.*

Невыполнение вышеуказанных требований влечет за собой снижение получаемых за задачи баллов

Задача №2.

Цифровое табло, состоящее из семи-сегментных дисплеев, используется для вывода числовых значений. Вы можете ***деактивировать уже включенные*** сегменты дисплея, но ***включать*** выключенные сегменты не можете. Необходимо написать программу или алгоритм на языках C, C++, Pascal для определения минимального числа, которое может быть выведено на дисплей и не будет меньше ***M***.



Входные данные:

Целое число и ограничение M. Количество дисплеев равно количеству цифр во введенном числе.

Выходные данные:

Целое число. Количество цифр в конечном варианте должно быть равно количеству цифр в начальном.

Пример:

Ввод: 86 10

Вывод: 15

****

Требования к оформлению задач по программированию:

5) *Программы должны быть написаны на одном из языков: C, C++, Pascal*
6) *Полностью оформленная задача должна содержать:*
   - *программу, выполняющую необходимые операции для всех допустимых данных;*
   - *операции с файлами входных и выходных данных **или** понятный пользователю интерфейс ввода исходных данных;*
   - *комментарии к тексту программы, облегчающие ее понимание.*

Невыполнение вышеуказанных требований влечет за собой снижение получаемых за задачи баллов

Задача №3.

Существует такая комната прямоугольной формы, с целочисленными длиной **l** и шириной **w**, для которых верно равенство **l = Aw + B**, где **A** и **B** – целочисленные постоянные. Количество единиц площади этой комнаты таково, что если прибавить к нему **C**, то результат будет делиться без остатка на простое число **P**. Найдите все возможные значения **w**.

Входные данные:

Первая строка содержит количество тестовых случаев, каждая следующая – тестовый случай с разделенными пробелом значениями **A**, **B**, **C** и **P**.

Выходные данные:

Для каждого случая с новой строки выводится результат решения, который содержит общее количество решений и полученные значения ширины в порядке возрастания. Все значения разделяются пробелом

Ограничения:

- $2 \le P < 10^6$
- $0 < A < P$
- $0 \le B$
- $C < P$

Пример:

Ввод:

```
2
1 1 0 2
1 2 2 3
```

Вывод:

```
2 0 1
0
```

****

Требования к оформлению задач по программированию:

7) *Программы должны быть написаны на одном из языков: C, C++, Pascal*
8) *Полностью оформленная задача должна содержать:*
   - *программу, выполняющую необходимые операции для всех допустимых данных;*
   - *операции с файлами входных и выходных данных **или** понятный пользователю интерфейс ввода исходных данных;*
   - *комментарии к тексту программы, облегчающие ее понимание.*

Невыполнение вышеуказанных требований влечет за собой снижение получаемых за задачи баллов

№1 Чистовик!

```
#include <vector>
#include <iostream>
#include <string> // язык C++11
using namespace std;
int main(){
    int f, d;
    cin >> f;
    string S;
    cin >> S >> d;
    vector <int> a(f, 0); // создаем массив из d элементов равных нулю,
    while (1){                // в которых мы будем перебирать слова
        int ok = 1; // флаг, который показывает, подходит ли нам слово
        int out = 1; // флаг, который показывает все ли числа мы перебрали
        vector <int> how(f, 0); // массив, который хранит сколько каких букв в слове
        for(int i=0; i<d; i++){ // проверка слова на повторы букв в нем
            how[a[i]]++;
            if(how[a[i]] > 3)
                ok = 0;
        }
        if (ok == 1){           // если слово подошло, выводим его
            for(int i=0; i<d; i++)
                cout << S[a[i]];
            cout << endl;
        }
        out=0;
        a[d-1]++;
        int i=d-1;
        while (a[i] == f){
            if(a[i]==f && i==0){
                a[i] =0;
                out =1;
            }
            else{
                a[i]=0;
                a[i-1]++;
                i--;
            }
        }
        if (out == 1) // если мы перебрали все возможные слова, то выходим
            break;
    }
    return 0;
}
```

создаем следующее слово для перебора

(6)

(1)

```
#include <iostream>  // язык C++11

using namespace std;

int main(){
    int n, m;
    cin >> n >> m;
    int a[10][10]; // массив, в которым мы будем смотреть, можно ли перейти от одного символа
    for(int i=0; i<10; ++i) // к другому, т. е. в a[i][j] хранится 1 если из i можно дойти j и 0 в противном
        for(int j=0; j<10; ++j){ // случае
            a[i][j]=0;
            if(i==j)
                a[i][j]=1;
        }
    a[0][1]=a[3][1]=a[0][7]=a[9][1]=a[6][5]=a[7][1]=a[9][1]=a[9][3]=a[9][4]=a[9][5]=a[9][7]=1;
    for(int i=0; i<10; ++i)
        a[8][i]=1;
    int step[20]; // массив хранит в себе степени 10 (a[i]=10^i)
    step[0]=1;
    int k=0; // кол-во цифр в n.
    int ans=0; // будем хранить ответ
    int n0=n;
    while(n0>0){
        n0/=10;
        k++;
        step[k]=10*step[k-1];
    }
    if(m>=step[k]){
        cout<<"Impossible"; // если в m больше цифр чем в n, то такое число
        return 0;           // нельзя получить
    }
    int flag=1; // флаг, который показывает равны ли части или, которые мы
                // прошли
    for(int i=0; i<k; i++){
        int st=0, // переменная, в которой хранится число, которое можно стереть на данном шаге
        if(flag==1)
            st=m/step[k-1-i];
        int now=n/step[k-1-i];
        int best=-1; // какая цифра лучше всего подходит сейчас
```

Чистовик!


```
    for(int j = st; j < 10; j++)
        if(best == -1 && a[now][j] == 1)
            best = j;
    if(best != st)
        flag = 0;
    if(best == -1){
        cout << "Impossible";
        return 0;
    }
    ans *= 10;
    ans += best;
    n %= step[k-1-i];
    m %= step[k-1-i];
}
cout << ans;
return 0;
}
```

№3

```
#include <iostream>    // язык C++11
#include <vector>

using namespace std;
int main(){
  int n;
  cin>>n;
  for(int i=0;i< n;i++){
    long long a, b, c, p;
    cin >> a >> b >> c >> p;   //T.K I= Aw+B [illegible] (Iw+c):p, [illegible](Aw+B)w+c):p [illegible](Aw²+Bw+c):p
    vector <long long> ans;
    for(int j=0; j < p; j++)
      if((a*j*j + b*j + c)% p==0)   //т.к. a,p≤10^6, то a*j*j ≤10^18, что не вызовет переполнение
        ans.push_back(j);          //переменной, т.к. это long long переменные.
    cout<< ans.size()<< endl;
    for(int j=0;j< ans.size();j++)
      cout<< ans[j] << " ";
    cout << endl;
  }
  return 0;
}
```

№4

```
#include <iostream>   // язык C++11
#include <vector>

using namespace std;
int n, m, color[2]; // размеры поля и кол-во вершин 2-х цветов в одной компоненте связности
vector<vector<char>> g(10000, vector<char>(10000)); // массив, в котором будут храниться поле
vector<vector<int>> used(10000, vector<int>(10000)); // массив, который показывает, заходили мы в вершину
void dfs(int x, int y, int pcolor){ // функция, которая будет идти по полю графу
pcolor = (pcolor+1)%2; // меняем цвет вершины относительно предыдущей
color[pcolor]++;
if(x>=2 && y>=2 && g[x-1][y-1]!='.' && g[x-2][y-2]=='F' && used[x-2][y-2]==0)
    dfs(x-2, y-2, pcolor);
if(x>=2 && y<m-2 && g[x-1][y+1]!='.' && g[x-2][y+2]=='F' && used[x-2][y+2]==0)
    dfs(x-2, y+2, pcolor);
if(x<n-2 && y<m-2 && g[x+1][y+1]!='.' && g[x+2][y+2]=='F' && used[x+2][y+2]==0)
    dfs(x+2, y+2, pcolor);
if(x<n-2 && y>=2 && g[x+1][y-1]!='.' && g[x+2][y-2]=='F' && used[x+2][y-2]==0)
    dfs(x+2, y-2, pcolor);
// этими 4-мя if-ами мы переходим функцией в клетки, в которые можно попасть с данной в клетке x, y.
}

int main(){
    cin >> n >> m;
    while(n>0 && m>0){
        for(int i=0; i<n; i++)
            for(int j=0; j<m; j++)
                cin >> g[i][j];
        for(int i=0; i<n; i++)
            for(int j=0; j<m; j++){
                if(g[i][j]=='G'){
                    if(i>=2 && j>=2 && g[i-2][j-2]=='F') g[i-2][j-2]='B';
                    if(i>=2 && j<m-2 && g[i-2][j+2]=='F') g[i-2][j+2]='B';
                    if(i<n-2 && j<m-2
```

4

чистовик!


```
for(int i=0; i<n; i++)
  for(int j=0; j<m; j++){
    if(p[i][j]=='d'){
      if(i>=2 && j>=2 && p[i-1][j-1] && p[i-1][j-1]!='P' && p[i-2][j-2]=='F')
        p[i-2][j-2]='B';
      if(i>=2 && j<m-2 && p[i-1][j+1]!='P' && p[i-2][j+2]=='F')
        p[i-2][j+2]='B';
      if(i<n-2 && j<m-2 && p[i+1][j+1]!='P' && p[i+2][j+2]=='F')
        p[i+2][j+2]='B';
      if(i<n-2 && j>=2 && p[i+1][j-1]!='P' && p[i+2][j-2]=='F')
        p[i+2][j-2]='B';
      // Этими 4-мя мы помечаем вершины, на которые
      // нельзя ставить башни, т.к. они будут под ударом
      used[i][j]=0; // обновляем массив used
    }
  int ans=0;
  color[0]=color[1]=0;
  for(int i=0; i<n; i++)
    for(int j=0; j<m; j++)
      if(used[i][j]==0 && p[i][j]=='F'){
        dfs(i,j,0); // таким образом мы проверим для
        ans+=max(color[0],color[1]); // всех свободных вершин,
        color[0]=color[1]=0;         // связанных между собой (т.е. до которых
      }                              // можно добраться прыжками на (±2; ±2) от i, j)
  cout<<ans<<endl;                   // от (i, j)
  cin>>n>>m;                         // и добавляем максимальную из color[0] и color[1], т.к
}                                    // граф двудольный
return 0;
}
```

(10)

(5)