Ключи к заданиям муниципального этапа всероссийской олимпиады школьников 2022/23 учебного года по информатике

Тула 2022

Задача 1. Выставляем!

Однажды Ксюша и Миша узнали, что в городе Квадратинске хотят провести Выставку научных достижений 2022 таким образом, чтобы путь до нее для всех жителей города был как можно короче. Все ***N*** домов города находятся на абсолютно ровной равнине. В Квадратинске нет обычных адресов, вместо них используются координаты домов в двумерном декартовом пространстве. Передвигаться по городу можно только таким образом, чтобы путь от здания $A(x_a,y_a)$ до здания $B(x_b, y_b)$ можно было определить по формуле $|x_a-x_b|+|y_a-y_b|$, где $|x|$ — модуль величины x.

Ребята заинтересовались вопросом: Сколько в Квадратинске существует возможных мест для размещения выставки? Учитывая, что выставка может быть размещена как в специально для нее построенном, так и в уже существующем доме, помогите Мише и Ксюше (а заодно и организаторам выставки) найти возможные места для размещения выставки. При этом следует знать, что чиновники города подают список адресов не по количеству домов, а по количеству жителей города, поэтому в одном и том же списке могут быть несколько раз упомянуты одни и те же дома.

Требуется написать программу вычисления количества возможных неповторяющихся мест для размещения выставки.

Технические требования:

Имя входного файла: *INPUT.TXT*

Имя выходного файла: *OUTPUT.TXT*

Ограничение по времени тестирования: 2 секунды на один тест.

Входные данные

В первой строке входного файла *INPUT.TXT* записано натуральное число ***N*** ($1\le N\le 2022$). В следующих ***N*** строках находятся координаты домов - 0 пары целых чисел (x_i,y_i) ($0\le x_i,y_i\le 10^9$).

Выходные данные

В выходном файле *OUTPUT.TXT* содержится единственное целое число – количество неповторяющихся мест для размещения выставки.

Примечание

Выставка может быть размещена в существующем доме или в доме, построенном специально для нее.

Примеры файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
4 0 1	8

0 2 3 2 3 1	
4 1 1 0 1 0 0 1 0	4
3 0 0 0 0 0 0	1

Решение

Для начала заметим, что задача не двумерная. Если мы меняем x координату точки, то сумма дистанций по y не меняется совсем. Так что нам нужно посчитать количество хороших точек на линии сначала с координатами x, потом с y и перемножить ответы.

Теперь, чтобы посчитать ответ на линии мы можем воспользоваться известным фактом: все точки с минимальным расстоянием лежат между левой и правой медианой. Так что теперь мы можем отсортировать массив и взять элементы на позициях $\left\lfloor\frac{n+1}{2}\right\rfloor$ и $\left\lfloor\frac{n+2}{2}\right\rfloor$ и вернуть их разность плюс один.

```
def solve(x):
    x.sort()
    return x[len(x) // 2] - x[(len(x) - 1) // 2] + 1

n = int(input())
x, y = [], []
for j in range(n):
    px, py = map(int, input().split())
    x.append(px)
    y.append(py)
print(solve(x) * solve(y))
```

Задача 2. Играем!

Ксюша и Миша играют в Легкие фишки. По правилам игры необходимо за минимальное количество ходов набрать определенную сумму очков из фишек. У ребят есть набор из неограниченного количества фишек весом от 1 до 5 очков. За каждый ход Ксюша и Миша должны выложить одну фишку любого веса. Помогите ребятам победить.

Требуется написать программу, определяющую минимальное количество фишек, необходимых для победы Ксюши и Миши.

Технические требования:

Имя входного файла: *INPUT.TXT*

Имя выходного файла: *OUTPUT.TXT*

Ограничение по времени тестирования: 2 секунды на один тест.

Входные данные

В первой строке входного файла *INPUT.TXT* записано единственное натуральное число $\boldsymbol{M}$ ($1 \leq \boldsymbol{M} \leq 1\,000\,000$) – число очков, которое нужно набрать ребятам.

Выходные данные

В выходном файле *OUTPUT.TXT* должно содержаться единственное целое число, равное минимальному количеству фишек, которые необходимо выложить Ксюше и Мише для победы.

Примеры файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
7	2
29	6
42	9

Решение

Оптимально делать наибольший возможный шаг каждый раз. Поэтому слоник должен сделать сначала некоторое количество шагов на расстояние 5, а затем один или ноль шагов на меньшее расстояние. Следовательно, ответ равен $\left\lceil \frac{x}{5} \right\rceil$.

```
#include <iostream>

using namespace std;

int main() {
    int m;
    cin >> m;
    cout << (m + 4) / 5 << '\n';
}
```

Задача 3. Работаем!

Ксюша и Миша однажды попали на склад Длинный и узнали, что на складе работают хитрые роботы-грузчики, которые хотят показать, что они очень нужны и постоянно заняты делом. Для этого они постоянно переносят небольшие коробки между пустыми стеллажами, и меняют на этих стеллажах одни пустые коробки на другие пустые коробки. На складе Длинный все стеллажи установлены в один ряд вдоль одной стены коридора с шагом в один метр (относительно центров стеллажей). Длина склада Длинный составляет 2022 метра. Роботы-грузчики могут разгружаться и загружаться только ровно напротив центра стеллажа и могут запомнить места только двух стеллажей за раз, а потому всю смену один робот-грузчик передвигается только между центрами одной пары стеллажей.

Ксюша и Миша заинтересовались работой роботов и решили составить маршруты роботов-грузчиков таким образом, чтобы все пустые стеллажи были включены в чей-то маршрут, а суммарная длина всех маршрутов была минимальна: хитрые роботы экономят заряд. Количество роботов не ограничено.

Требуется написать программу, которая вычисляет минимальную суммарную длину маршрутов роботов для заданного количества стеллажей.

Технические требования:

Имя входного файла: *INPUT.TXT*

Имя выходного файла: *OUTPUT.TXT*

Ограничение по времени тестирования: 2 секунды на один тест.

Входные данные

В первой строке входного файла *INPUT.TXT* записано натуральное число N – количество пустых стеллажей ($2 \le N \le 10^9$). Во второй строке записано N чисел – координаты стеллажей в метрах относительно входа на склад, каждое число является натуральным числом не большим, чем 2022.

Выходные данные

В выходной файл *OUTPUT.TXT* необходимо вывести единственное натуральное число – минимальную суммарную длину маршрутов роботов.

Примеры файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
7 1 2 3 10 27 14 21	12
5 5 11 0 15 3	9
3 10 0 99	12

Решение 1.

Отсортируем координаты стеллажей в массиве a. Пусть dp[i] равно минимальной суммарной длине всех маршрутов, когда любые два стеллажа от первого (нумерация стеллажей начинается с 1) до i-го включены в маршрут.

При n = 2 оба стеллажа обязательно должны быть включены в маршрут:

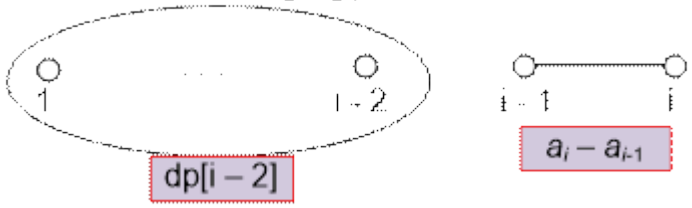
dp[2] = a2 – a1

При n = 3 следует включить первый стеллаж и второй в один маршрут, а второй с третьим в следующий. Таким образом

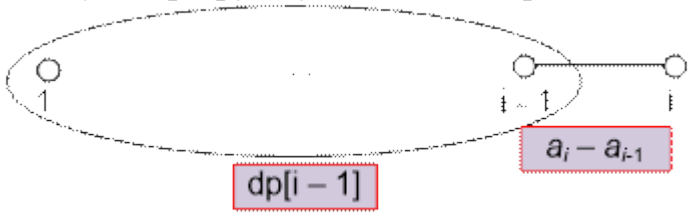
dp[3] = a3 – a1

При добавлении i-го стеллажа имеются две возможности присоединения его в маршрут:

1) соединяем первые i – 2 стеллажей между собой, а (i – 1)-ый стеллаж соединяем с i-ым. Общая длина маршрута для такого соединения составит dp[i – 2] + a[i] – a[i – 1].



2) соединяем первые i – 1 стеллажей между собой, а i-ый стеллаж подсоединяем к (i – 1)-ому. Маршрут будет длиной dp[i – 1] + a[i] – a[i – 1].



Выбираем тот метод соединения, при котором общая длина маршрута наименьшая. То есть

dp[i] = min(dp[i – 2], dp[i – 1]) + a[i] – a[i – 1]

Решение 2.

```
#include <bits/stdc++.h>

using namespace std;

int main()
{
  int n, bb;
  cin >> n;
  vector<int> a;
  vector<int> s;
  for(int i = 0; i < n; ++i)
  {
    cin >> bb;
    a.push_back(bb);
  }
  sort(a.begin(),a.end());
  if(n == 2)
  {
    cout << a[1] + a[0];
    return 0;
  }
```

```
  s.push_back(a[1] - a[0]);
  s.push_back(a[2] - a[0]);
  for(int i = 2; i < a.size() - 1; ++i)
    s.push_back(min(s[i-1],s[i-2]) + abs(a[i] - a[i+1]));

  cout << s[s.size() - 1];
  return 0;
}
```

Задача 4. Включаем/выключаем!

Однажды Ксюша и Миша приехали на экскурсию на фирму «Переключалкино». Директор фирмы рассказал ребятам об интересной задаче, которую ему пришлось решить совсем недавно. Эта фирма переехала в новый офис, в котором заняла ровно N разных кабинетов. При переезде свет в некоторых кабинетах был отключен, а в других – включен. Однако, работать в офисе можно, только если свет включен во всех комнатах одновременно. Электрощит офиса имеет M двухфазных переключателей. От фазы каждого переключателя зависит состояние света в определенных кабинетах. Точно известно, что свет в любом из кабинетов переключается ровно двумя переключателями. Известно состояние подачи света в каждый кабинет. При изменении фазы переключателя меняется состояние подачи света в те кабинеты, за которые отвечает этот переключатель (т.е. если свет в кабинете был включен, то он выключается и наоборот). К примеру, если в 1, 2 и 3 кабинетах состояние света описывается константами «включен», «включен» и «выключен», то после смены фазы переключателя, связанного с этими кабинетами, состояние изменится на «выключен», «выключен» и «включен» соответственно. Директор смог начать работу в офисе, а ребята заинтересовались этой занимательной задачей.

Требуется написать программу, которая определяет по начальному состоянию света во всех кабинетах, можно ли начать работу в офисе или это невозможно и день следует объявить выходным.

Технические требования:

Имя входного файла: *INPUT.TXT*

Имя выходного файла: *OUTPUT.TXT*

Ограничение по времени тестирования: 2 секунды на один тест.

Формат входных данных

В первой строке входного файла *INPUT.TXT* записаны два натуральных числа N – число кабинетов ($2 \le N \le 10^5$), и M – число переключателей ($2 \le M \le 10^5$).

Во второй строке записано N чисел, обозначающих состояние света в соответствующем кабинете на момент приезда фирмы в новый офис. Состояние характеризуется константами 1 – свет выключен, 0 – свет включен.

Далее следуют M строк, каждая из которых начинается с числа K ($2 \le K \le N$), задающего количество кабинетов, связанных с i-ым переключателем ($1 \le i \le M$), после которого перечисляется K номеров кабинетов X_i ($2 \le X_i \le N$). В качестве номеров кабинетов используются неповторяющиеся натуральные числа от 1 до N. В каждом кабинете свет переключается ровно двумя переключателями.

Формат выходные данные

В единственной строке выходного файла содержится слово WORK (без кавычек), если можно дать свет во все кабинеты одновременно, иначе содержится слово WEEKEND (без кавычек).

Примеры файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
3 3 1 0 1 2 1 3 2 1 2	WEEKEND

2 2 3	
3 3 1 0 1 3 1 2 3 1 2 2 1 3	WORK
3 3 1 0 1 3 1 2 3 2 1 2 1 3	WEEKEND

Решение

Смоделируем ситуацию в виде графа с кабинетами в качестве ребер и переключателями в качестве узлов.

Все кабинеты представлены в виде ребер. Отметьте ребра как "1", если свет есть, в противном случае отметьте ребро как "0". Ответ будет УТВЕРДИТЕЛЬНЫМ, если вы сможете раскрасить граф таким образом, чтобы ребра, имеющие значение 0, имели оба узла разного цвета, а ребра, имеющие значение 1, имели оба узла одного цвета. Для проверки достаточно использовать алгоритм BFS с сменой фазы переключателей.

```
#include <bits/stdc++.h>
using namespace std;
#define mem(x,y) memset(x,y,sizeof(x))
#define bitcount __builtin_popcountll
#define mod 1000000007
#define N 1000009
#define fast ios::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define ss(s) cin>>s;
#define si(x) scanf("%d",&x);
#define sl(x) cin>>x;
#define pb push_back
#define mp make_pair
#define all(v) v.begin(),v.end()
#define S second
#define F first
#define ll long long
int n,m;
vector<int> arr[N];
int arr1[N];
vector<pair<int,int > > adj[100008];     //max-size
void graph(int n)
{
  for (int i=1; i<=n; i++)
  {
    int x = arr[i][0];
    int y = arr[i][1];
    adj[x].push_back(mp(y,arr1[i]));  //arr1[i] stores th color of the edge
    adj[y].push_back(mp(x,arr1[i]));
  }
}
```

```
int col[N];
bool vis[N];

int main()
{
  cin>>n>>m;
  assert(n>=2 && n<=100000);
  assert(m>=2 && m<=100000);
  for (int i =1; i<=n; i++)
    cin >> arr1[i];
  for (int i = 1; i<=m; i++)
  {
    int sz;
    cin >> sz;
    for (int j =0; j<sz; j++)
    {
      int y;
      cin >> y;
      arr[y].push_back(i);
    }
  }
  graph(n);
  mem(col,-1);
  mem(vis,false);
  bool p = true;
  for (int i = 1;i<=m;i++)
  {
    if (!vis[i] && adj[i].size())
    {
      col[i] = 0;
      vis[i] = true;
      queue <int > q;
      q.push(i);
      while(q.size())
      {
        int node = q.front();
        q.pop();
        for (auto pi:adj[node])
        {
          int co = col[node];
          if (pi.S == 0)
          {              //if the edge is 0 then give the opposite color.
           co = 1 - co;
          }
          if(vis[pi.F] && col[pi.F] != co)
          {
            p = false;
          }
          if (!vis[pi.F])
          {
            col[pi.F] = co;
            vis[pi.F] = true;
            q.push(pi.F);
          }
        }
```

```
      }
    }
  }
  if (!p)
    cout<<" WEEKEND ";
  else
    cout<<"WORKDAY";
}
```

Задача 5. Вычеркиваем!

Миша предлагает Ксюше решить очень интересную задачку. Он записывает натуральное число ***N***, состоящее из ***K*** цифр, и натуральное число ***M*** (***K***>***M***). Ксюша должна определить, какие ***M*** цифр нужно вычеркнуть из записи числа ***K***, чтобы оставшиеся цифры составляли наименьшее число. В качестве ответа Ксюша должна выдать в порядке возрастания номера вычеркнутых цифр (цифры в заданном числе нумеруются слева направо, начиная с единицы).

Требуется написать программу, которая поможет Ксюше справится с задачкой Миши.

Технические требования:

Имя входного файла: *INPUT.TXT*

Имя выходного файла: *OUTPUT.TXT*

Ограничение по времени тестирования: 2 секунды на один тест.

Формат входных данных:

Входной файл *INPUT.TXT* состоит из двух строк.

В первой строке содержится натуральное число ***N*** (***N***$\leq 10^{250}$).

Во второй строке содержится натуральное число ***M*** (***M***<***N***), задающее количество вычеркиваемых цифр.

Формат выходных данных:

Выходной файл *OUTPUT.TXT* содержит упорядоченную по возрастанию последовательность номеров ***M*** цифр, которые нужно вычеркнуть из заданного числа (цифры в заданном числе нумеруются слева направо, начиная с единицы).

Пример файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
462432 2	1 2
672591831 4	1 2 5 7
111462711492 2	7 11

Решение

Считываем натуральное число в строку `stroka`. Создаем массив `otvet`, элементами которого являются состояния для букв исходной строки: ***0*** – буква не вычеркнута, ***1*** – буква вычеркнута. В начале все элементы массива равны нулю.

Рассматриваем все подстроки исходной строки, которые

1. Начинаются с наименьшего возможного для рассмотрения индекса.
2. Имеют длину ***K-M.***

На каждой итерации рассмотрения ищем максимальную цифру, входящую в подстроку, и вычеркиваем ее с позиции наименьшего индекса вхождения. При этом делаем отметку о вычеркивании в массиве ответов.

Для формирования ответа используем данные массива `otvet`**.**

```
Program task_vicherkivaem;
  uses crt;
  var          m, i, j, gran, max, ind_max, zi, code: longint;
               stroka : string;
               otvet  :  array [1..250] of byte;
begin
  for i:=1 to 250 do otvet[i]:=0;
  i:=1;
  readln(stroka);
  readln(m);
  gran:=length(stroka)-m;
  while (i<=m) do
     begin
     val(stroka[1], zi, code);
     max:=zi; ind_max:=1; j:=1;
     while (j<=gran) do
       begin
       val(stroka[j], zi, code);
       if (zi>max) then
            begin
            max:=zi; ind_max:=j;
            end;
       j:=j+1;
       end;
    stroka[ind_max]:='0';
    otvet[ind_max]:=1;
    gran:=gran+1;
    i:=i+1;
   end;
 //writeln('Otvet');
 for i:=1 to length(stroka) do
    if (otvet[i]=1) then write( i, '  ' );
 end.
```

Задача 6. Находим!

Ксюша предлагает Мише решить очень интересную задачку. Она записывает натуральное число N и предлагает Мише найти два натуральных числа M и K, для которых у чисел A^M и A^K совпадают последние две цифры.

Требуется написать программу, которая поможет Мише справится с задачкой Ксюши.

Технические требования:

Имя входного файла: *INPUT.TXT*

Имя выходного файла: *OUTPUT.TXT*

Ограничение по времени тестирования: 2 секунды на один тест.

Формат входных данных:

Входной файл *INPUT.TXT* содержит единственное натуральное число A ($2 \le A \le 1000$).

Формат выходных данных:

Выходной файл *OUTPUT.TXT* содержит два записанных через пробел натуральных числа M и K, удовлетворяющих условию задачи.

Пример файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
5	2 3
36	2 7
44	2 12

Решение

Заводим массив для контроля двух последних цифр очередной степени введенного числа. Первым элементом массива будет число, состоящее из двух последних цифр той степени введенного числа, которая больше 10, это и будет число M. Далее при формировании очередного элемента массива определяем, входит он в массив или нет. Первое совпадение по индексу соответствует число K.

```
Program task;
  const n = 100;
  type mas = array [1..n] of integer;
  var a,m,k,i,j,x:integer; v:mas; f,ff:boolean;
Begin
  //writeln('Введите число a: ');
  readln(a);
  m:=1; //первое требуемое число
  x:=a; //временная переменная для вычисления последних двух цифр
оцередной степени исходного числа
  while x<10 do //пока в числе меньше двух цифр
  begin
    x:=x*a;
    m:=m+1
  end;
  f:=1=1; //признак найденного второго числа
  i:=2;
```

```
  v[1]:=x mod 100;
  v[2]:=v[1]*a mod 100;
  while (v[i-1]<>v[i]) and f do
  begin
   j:=1;
   ff:=1=1;
   i:=i+1;
   x:=(v[i-1]*a) mod 100;
   while (j<i) and ff do
     if x<>v[j] then j:=j+1
     else ff:=1=2;
   if ff then
     v[i]:=x
   else f:=1=2;
  end;
  if m=1 then
    k:=i //второе требуемое число
  else
    k:=m+i-1;
  write( m, ' ', k);
end.
```

Задача 7. Вычисляем!

Ксюша и Миша пытаются вычислить уникальность предлагаемого им слова, состоящего из букв английского алфавита и цифр. При этом, под уникальностью слова, согласно определению их учителя Ильи Юрьевича, они понимают количество букв, встречающихся в слове ровно один раз. Например, уникальность слова «INFORMATICS» равна 9 т.к. буквы N, F, O, R, M, A, T, C, S входят в это слово ровно по одному разу, а уникальность слова «Informatics» равна 11, т.к. каждая буква входит в это слово ровно по одному разу, а уникальность строки «22222» равна нулю.

Требуется написать программу, которая поможет Ксюше и Мише посчитать уникальность предлагаемого слова.

Технические требования:

Имя входного файла: *INPUT.TXT*

Имя выходного файла: *OUTPUT.TXT*

Ограничение по времени тестирования: 2 секунды на один тест.

Формат входных данных:

Входной файл *INPUT.TXT* содержит единственное слово, составленное из заглавных и строчных букв английского алфавита и цифр. Длина слова не превосходит 20000 символов.

Формат выходных данных:

Выходной файл *OUTPUT.TXT* содержит единственное целое неотрицательное число, равное уникальности введенного слова.

Пример файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
INFORMATICS	9
Informatics	11
Thefiveboxingwizardsjumpquickly	23
12345678901234567890123456789012345678901234567890	0

Решение

Заведем массив из 256 элементов и обнулим его. Далее проходим по каждому символу заданное слова и по коду символа увеличиваем значение того элемента массива, индекс которого совпадает с кодом. Для вычисления уникальности проходим по массиву и считаем количество тех элементов, значение которых равно 1.

```
Program task;
  const n = 255;
  type mas = array [0..n] of integer;
  var k,i,j:integer; v:mas; str:string;
Begin
  for i:=0 to 255 do
    v[i]:=0;
  //writeln('Введите слово: ');
  readln(str); //ограничение на длину строки в 256 символов можно
обойти, если считывать каждый символ из файла
  for i:=1 to length(str) do
    v[ord(str[i])]:=v[ord(str[i])]+1;
```

```
  k:=0;
  for i:=0 to 255 do
    if v[i]=1 then k:=k+1;
  //writeln('Уникальность слова = ', k);
  write(k);
end.
```

Задача 8.Отдыхаем!

Ксюша и Миша на каникулах пошли в зоопарк. Там они увидели прекрасных розовых фламинго и очень заинтересовались их способностью долго стоять на одной ноге, не теряя равновесия. В птичьем домике находились несколько фламинго, одни из которых стояли на одной ноге, а другие – на двух. В домике была открыта только нижняя половинка двери, через которую были видны только ноги фламинго. Миша решил посчитать все видимые ноги фламинго, и у него получилось число ***N***. Затем к Мише подошла Ксюша и тоже посчитала количество видимых ног. У Ксюши получилось число ***M***, поскольку одни фламинго успели опустить вторую ногу, другие – поднять, а третьи стояли неподвижно. Ребята захотели узнать, сколько же фламинго находится в домике? Поразмыслив хорошенько, Ксюша и Миша поняли, что по ногам определить количество фламинго можно не во всех случаях. Тогда они попытались определить минимальное и максимальное возможное количество фламинго в птичьем домике по двум полученным числам их ног.

Требуется написать программу, которая поможет Ксюше и Мише посчитать количество фламинго в птичьем домике по двум заданным числам их ног в разные промежутки времени.

Технические требования:

Имя входного файла: *INPUT.TXT*

Имя выходного файла: *OUTPUT.TXT*

Ограничение по времени тестирования: 2 секунды на один тест.

Формат входных данных:

Входной файл *INPUT.TXT* содержит два натуральных числа ***N*** и ***M***, разделенных пробелом ($1 \le \boldsymbol{N} \le 10^9$, $1 \le \boldsymbol{M} \le 10^9$)

Формат выходных данных:

Выходной файл *OUTPUT.TXT* два натуральных числа, разделенных пробелом – минимальное и максимальное количество фламинго, которые могли быть в птичьем домике.

Примеры файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
3 4	2 3

Решение

Максимальное количество розовых фламинго определяется по минимальному из двух чисел (каждая птица стоит на одной ноге), а минимальное – как целое число, вдвое меньшее минимального (учитываем, что для четного и нечетного максимального будут разные ответы). В случае, если большее из чисел ***N*** и ***M*** превосходит удвоенное максимальное, то задача не имеет решений.

```
Program task;
  uses crt;
  var  {t,t1:text;}
     a,b, max, min, maxab : longint;
function minim ( x,y: longint): longint;
begin
```

```
          if (x<y) then minim:=x else minim:=y;
     end;
     function maximum ( x,y: longint): longint;
     begin
          if (x>y) then maximum:=x else maximum:=y;
     end;

     begin
       {assign(t,'input.txt');   assign(t1,'output.txt');
       reset(t);   rewrite(t1);
       read(t,a);
       read(t,b);
       close(t);}

       read(a);
       read(b);
       max:= minim(a,b);
       maxab:=  maximum ( a,b);
       if (max mod 2 = 0) then min := max div 2
          else min := max div 2+1;
       if maxab <= 2*max then write(min:10, max:10) else write
('reshenij net');
       {close(t1);}
     end.
```