Задача 1. Матрешки

Медведь решил начать делать матрешки. Стенки матрешек у Медведя получились по 5 мм. В порыве энтузиазма Медведь не заметил как заполнил весь цех матрешками. Размеры всех матрешек сразу после их создания Медведь записывал в список готовых изделий, но забыл расставить их по порядку. Помогите Медведю собрать самый большой по количеству набор матрешек по правилу, что в одной матрешке может быть любая другая матрешка размером меньше ее как минимум на 2мм.

Требуется написать программу вычисления количества матрешек в самом большом по количеству матрешек наборе, который можно получить из имеющегося в наличии.

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 ($1 < N \leq 2023$). В следующих N строках находятся размеры наличных в цеху матрешек. Размер матрешки — число M – указан в миллиметрах, $50 \leq M \leq 10000$.

Выходные данные

В выходном файле *OUTPUT.TXT* содержится единственное целое число – количество матрешек в самом большом наборе.

Примеры файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
5 52 56 60 53 64	4
5 63 65 62 51 50	3

Система оценки.

Для оценки правильности решения предоставлено 10 пар файлов INPUT-ANSWER с эталонными входными и выходными данными

Каждый успешно пройденный тест — 10 баллов.

Решение:

Первым шагом решения задачи является упорядочивание последовательности чисел, далее отбираются только те контейнеры что удовлетворяют условию включения в последовательность. Очевидным образом, начинать формировать последовательность контейнеров лучше всего с наибольшего контейнера, так как если существует последовательность контейнеров без данного контейнера, то либо эту последовательность можно вложить в рассматриваемый контейнер, либо рассматриваемый контейнер может заменить самый большой контейнер в существующей последовательности.

```
f = open('INPUT.TXT')
n = int(f.readline())
a = [int(s) for s in f]
a.sort(reverse = True)

count = 1
s = a[0]
for i in range(1, len(a)):
     if s - a[i] >= 2:
          count += 1
          s = a[i]
print(count)
```

Задача 2. Проектная деятельность

Медведь решил привлечь к своей проектной деятельности новых участников, однако оказалось, что в лесной школе уже активно ведется деятельность по многим проектам и зверята не спешат присоединятся к команде нового проекта. В итоге Медведь определил что *i*-й зверенок вступит в проект, если в нем уже есть не менее A_i, но и не более B_i участников. В то же время Медведя успокоило, что вступив в команду проекта зверята уже не выйдут из него до конца проекта. Помогите Медведю составить правильную стратегию приглашения новых участников проекта. Медведя как руководителя проекта **не считаем** одним из участников. Медведь категоричен, и готов закрыть проект, если у него не получится вовлечь всех школьников.

Требуется написать программу определения последовательности приглашения новых участников в проект, где имена школьников будут закодированы их номерами.

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 ($1 \le N \le 2023$). В следующих N строках находятся пары чисел A_i и B_i через пробел, где $0 \le A, B \le 1000$

Выходные данные

В выходном файле *OUTPUT.TXT* содержится одну строку — последовательность номеров участников разделенных пробелом, в которой стоит приглашать новых участников. Если пригласить всех из представленного перечня невозможно, то в строке должно быть только одно число 0.

Примечание

Если возможно составить несколько разных последовательностей включения новых участников проекта, в качестве ответа приведите любой из них.

Примеры файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
5 2 3 0 2 0 4 1 2 3 4	2 4 1 3 5
7 2 3 1 3 0 4 1 2 3 4 5 6 6 6	3 4 1 2 5 6 7
7 2 3 1 3 0 4 1 2 3 4 4 4 5 6	0

Система оценки.

Для оценки правильности решения предоставлено 15 пар файлов INPUT-ANSWER с эталонными входными и выходными данными

Каждый успешно пройденный тест с номерами 1-10 — 5 баллов.

Каждый успешно пройденный тест с номерами 11-15 — 10 баллов.

Обратите внимание, что посимвольное сравнение результата работы тестируемого приложения и эталонного ответа неуместно.

Решение:

```
const nmax=100;

var a:array[1..nmax,1..2] of integer;
    b,c:array[0..nmax] of integer;
    n:integer;

procedure init;
  var i,j:integer;
  begin
  assign(input,'input.txt');
  reset(input);
  assign(output,'output.txt');
  rewrite(output);
  read(n);
  for i:=1 to n do
    for j:=1 to 2 do
      read(a[i,j]);
  for i:=1 to nmax do
    c[i]:=0;
  end;

procedure work;
  var i,j,k:integer;
  begin
  for i:=0 to n-1 do begin
    k:=0;
    for j:=1 to n do
      if (c[j]=0) and (a[j,1]<=i) and (a[j,2]>=i) then begin
        if k=0 then k:=j
        else if a[j,2]<a[k,2] then k:=j;
        end;
    if k=0 then begin
      writeln(0);
      exit;
      end;
    c[k]:=1;
    b[i]:=k;
    end;

  for i:=0 to n-2 do
    write(b[i],' ');
  write(b[n-1]);
  end;
```

```
procedure done;
  begin
  close(input);
  close(output);
  end;


begin
init;
work;
done;
end.
```

Задача 3. Набираем бонусы

Медведь решил немного отдохнуть и поиграть в популярную игру Бродилка. В игре персонаж находится в лабиринте из N комнат, каждая комната соединена с другими K комнатами коридорами, в которых расположены бонусы, для каждого коридора количество бонусов L_i разное. Коридоры позволяют двигаться только в одну сторону. Вход в лабиринт находится в комнате под номером 1, а выход в комнате с номером N. Постройте такой маршрут Медведь, чтобы он набрал как максимально возможное количество бонусов.

Обратим внимание, что коридоры могут быть закольцованы на самой комнате, т.е. вести в ту же комнату из которой начались, а так же из одной комнаты в другую комнату может быть одновременно несколько прямых коридоров с разным количеством бонусов в них. Также важно что бонусы можно не только накапливать, но и терять, если сумма накопления является отрицательным числом.

Требуется написать программу вычисления с максимально возможным количеством бонусов, включая вариант с бесконечным накоплением бонусов

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* записаны через пробел натуральные числа N и K – количество комнат и коридоров $(1 \le N \le 2023)$, $(1 \le K \le 20230)$. В следующих K строках находятся тройки чисел разделенные пробелом: первая пара чисел - номера комнат, из которой и в которую ведет коридор, последнее число в строки — это L_i количество бонусов, которое можно собрать в коридоре, где $-10000 \le L_i \le 10000$.

Выходные данные

В выходном файле *OUTPUT.TXT* содержится одну строку:

SUPER – если можно набрать неограниченное количество бонусов;

UNREAL – если игру пройти невозможно;

в остальных случаях записано число максимально возможно набранных бонусов.

Примечание

Коридоры могут быть закольцованы на самой комнате, т.е. вести в ту же комнату из которой начались, а так же из одной комнаты в другую комнату может быть одновременно несколько прямых коридоров с разным количеством бонусов в них.

Бонусы можно не только накапливать, но и терять, если сумма накопления является отрицательным числом.

Примеры файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
2 2 1 2 3 1 2 7	7
5 5 1 2 -3 2 3 2 3 4 2 4 2 2 2 5 -5	SUPER
5 5 1 2 -3 2 3 2 3 4 2 4 2 2 2 1 -5	UNREAL

Система оценки.

Для оценки правильности решения предоставлено 20 пар файлов INPUT-ANSWER с эталонными входными и выходными данными

Каждый успешно пройденный тест — 5 баллов.

Решение:

```
#include<bits/stdc++.h>
#define int long long
#define inf 1e10l
using namespace std;


struct edge { int u, v, c; };
vector<pair<int, int>> gp[2023];
vector<edge> g;
```

```
    vector<int> d(2023, -inf), p(2023, -1), used(2023, 0), ban(2023,
0);

    void bfs(int i, int c) {
        used[i] = c;
        for (auto& e : gp[i]) {
            if (!used[e.first]) bfs(e.first, c);
        }
    }

    int32_t main() {
        ifstream cin("input.txt");
        int n, m; cin >> n >> m;

        for (int i = 0; i < m; i++) {
            edge ins; cin >> ins.u >> ins.v >> ins.c;
            ins.u--; ins.v--;
            gp[ins.v].push_back(make_pair(ins.u, ins.c));
            g.push_back(ins);
        }
        d[0] = 0;
        for (int k = 0; k < n; k++) {
            for (auto& e : g) {
                if (d[e.u] > -inf && d[e.v] < d[e.u] + e.c) {
                    d[e.v] = d[e.u] + e.c;
                    p[e.v] = e.u;
                }
            }
        }

        if (d[n - 1] == -inf) {
            cout << "UNREAL";
            return 0;
        }
        bfs(n - 1, 1);

        int u = 2023;
        while (u != -1) {
            u = -1;
            for (auto& e : g) {
                if (!ban[e.u] && !ban[e.v]) {
                    if (d[e.u] > -inf && d[e.v] < d[e.u] + e.c) {
                        d[e.v] = d[e.u] + e.c;
                        p[e.v] = e.u;
                        u = e.u;
                    }
                }
            }
            if (u == -1) continue;
            vector<int> cyc;
            int y = u;
            for (int i = 0; i < n; i++) {
                y = p[y];
            }
```

```
            for (int cur = y; ; cur = p[cur]) {
                cyc.push_back(cur);
                if (cur == y && cyc.size() > 1)  break;
            }
            for (auto b : cyc) ban[b] = 1;
        }

        for (int i = 0; i < n; i++) {
            if (used[i] && ban[i]) {
                cout << "SUPER"; return 0;
            }
        }
        cout << d[n - 1];
        return 0;
}
```

Задача 4. Простой архиватор

Медведь решил создать собственный архиватор для текстов только на русском языке. Медведь задал несколько последовательность символов, которые будет заменять буквами английского алфавита. Медведь заметил, что некоторые тексты имеют несколько вариантов замены выбранных последовательностей на буквы английского алфавита. Помогите Медведю подобрать алгоритм сжатия таким образом чтобы в итоге получался как можно более короткий текст.

Требуется написать программу определения наилучшей комбинации замены последовательностей символов на буква английского алфавита по степени сжатия конечного текста.

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* записаны через пробел натуральные числа $\boldsymbol{N}$ и $\boldsymbol{K}$ – количество последовательностей символов на замену и количество строк в сжимаемом тексте, где $(1 \leq \boldsymbol{N} \leq 52)$, $(1 \leq \boldsymbol{K} \leq 2023)$. В следующих $2\boldsymbol{N}$ строках записаны на первой строке последовательность символов на замену (последовательность не включает в себя символ конца строки) вторая строка - и символ на замену. В следующих N строках записан текст для сжатия.

Выходные данные

В выходном файле *OUTPUT.TXT* содержится одну строку — последовательность номеров участников разделенных пробелом, в которой стоит приглашать новых участников. Если пригласить всех из представленного перечня невозможно, то в строке должно быть только одно число 0.

Примечание

Медведь использует регистрозависмое решение, поэтому большая (прописная) и малая (строчная) буквы не равносильны друг другу как в заменяемой последовательности, так и в подставляемых символах.

В сжимаемом тексте нет букв английского алфавита.

В заменяемых последовательностях нет букв английского алфавита.

Длина строки не может превышать 255 символов

В примере темным фоном отмечен последний символ строки в сжимаемом и в сжатом текстах.

Примеры файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
6 3 и F B L информат F аспекте h включает f себя g Д.С. Чернавский даёт следующее определение информатики: <Информатика - наука о процессах передачи, возникновения, рецепции, хранения и обработки информации>. Он предлагает выделять в информатике три направления: техническое, прикладное и фундаментальное. В техническом аспекте информатика включает в себя передачу, кодирование и приём информации. В прикладном аспекте информатика занимается разработкой компьютеров, созданием программ	Д.С. Чернавский даёт следующее определение Фики: <Информатика - наука о процессах передачи, возникновения, рецепции, хранения и обработки информации>. Он предлагает выделять в Фике три направления: техническое, прикладное и фундаментальное. В техническом h Фика f в g передачу, кодирование и приём информации. В прикладном h Фика занимается разработкой компьютеров, созданием программ (computer science). Фундаментальный аспект Фики f в g изуч ение процессов возникновения, эволюции, извлечения и реализации ценной информации.

(computer science). Фундаментальный аспект информатики включает в себя изуч ение процессов возникновения, эволюции, извлечения и реализации ценной информации.	

Система оценки.

Для оценки правильности решения предоставлено 5 пар файлов INPUT-ANSWER с эталонными входными и выходными данными

Каждый успешно пройденный тест — 20 баллов.

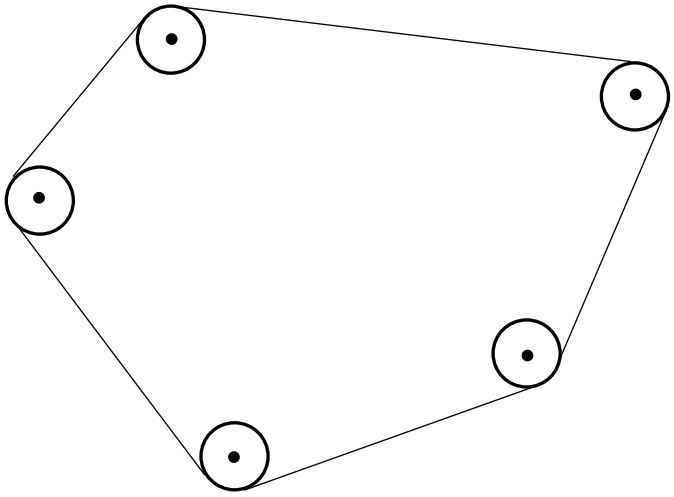
Решение:

```
const rmax=52;

var n,r:integer;
    p:array[1..rmax] of record
      c:char;
      s:string;
      end;
    a:array[0..255] of byte;
    b:array[0..255] of byte;
    tnum:integer;
    s:string;

procedure init;
  var i:integer;
      c:char;
  begin
  assign(input,'input.txt');
  reset(input);
  assign(output,'output.txt');
  rewrite(output);
  readln(r,n);
  for i:=1 to r do begin
    readln(p[i].s);
    readln(p[i].c);
    end;
  end;

procedure work;
  var i,j:byte;
      q:string;
      w:byte;
  begin
  //fillchar(a,sizeof(a),0);
  for i:=1 to 255 do
```

```
    a[i]:=0;
  a[0]:=1;
  b[0]:=0;
  for i:=1 to length(s) do begin
    a[i]:=a[i-1]+1;
    b[i]:=0;
    for j:=1 to r do begin
      w:=length(p[j].s);
      if w<=i then
        if a[i-w]+1<a[i] then
          if copy(s,i-w+1,w)=p[j].s then begin
             a[i]:=a[i-w]+1;
             b[i]:=j;
             end;
      end;
    end;
  q:='';
  i:=length(s);
  while i<>0 do
    if b[i]=0 then begin
       q:=s[i]+q;
       dec(i);
       end
    else begin
       q:=p[b[i]].c+q;
       i:=i-length(p[b[i]].s);
       end;
  writeln(q);
  end;


begin
init;
for tnum:=1 to n do begin
  readln(s);
  work;
  end;
close(input);
close(output);
end.
```

Задача 5. Моя территория!

Медведь решил огородить свой участок леса толстой проволокой, чтобы всякие Волки без надобности не рыскали где попало. Для этого он выбрал *K* надежных деревьев одного диаметра, образующих выпуклый многоугольник. Медведь был очень рачительным хозяином и в душе математиком, поэтому он решил рассчитать длину проволоки, которая ему понадобится. Для этого он начертил координатную плоскость, нанес на нее координаты центров окружностей, являющихся проекциями каждого выбранного деревца на плоскость земли. Так как диаметры деревьев были абсолютно равными, Медведь построил *K* окружностей одинакового радиуса из каждого центра и соединил их отрезками так, чтобы получился замкнутый контур. Ну вот примерно, как на рисунке:



А потом Медведь так перетрудился, что позабыл все формулы.

Требуется написать программу, которая поможет Медведю правильно вычислить длину толстой проволоки, которой нужно будет огородить его территорию.

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 нескольких строк.

В первой строке содержится натуральное число *K* ($1 < K < 200$).

Во второй строке содержится единственное натуральное число *R* – радиус ствола дерева.

В следующих *K* строках содержится *K* пар вещественных чисел, задающих координаты центров проекций деревьев на плоскость, построенную Медведем. Каждая координата не превосходит по модулю числа 1000. Пары координат перечисляются в порядке обхода по территории Медведя по часовой стрелке или против нее. Никакие два дерева не касаются друг друга.

Формат выходных данных:

Выходной файл *OUTPUT.TXT* содержит единственное вещественное число, округленное до двух знаков после запятой, равное длине проволоки, необходимой для огораживания территории Медведя.

Пример файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
4 1 0.0 0.0 2.0 0.0 2.0 2.0 0.0 2.0	14.28

Система оценки.

Для оценки правильности решения предоставлено 5 пар файлов INPUT-ANSWER с эталонными входными и выходными данными

Каждый успешно пройденный тест — 20 баллов.

Решение:

Опишем функцию вычисления длины отрезка по координатам его концов. Считываем первую пару координат и запоминаем ее для того, чтобы вернуться к нем в самом конце в силу замкнутости контура. В искомую длину вначале записываем длину окружности проекции ствола дерева на плоскость. Затем в цикле считываем каждую из оставшихся пар координат, считаем длину отрезка и переопределяем начало следующего отрезка. После выхода из цикла дописываем в результат длину последнего отрезка замкнутого контура.

```
Program task01;
  var k,r,i,j: integer;
      x0,y0,x,y,a,b:real;
      len:real; ft:text;

  function dlina(x1,y1,x2,y2:real):real;
  begin
    dlina:= sqrt((x1-x2)*(x1-x2)+ (y1-y2)*(y1-y2));
  end;

Begin
  assign(ft,'derevo.txt');
  reset(ft);
  readln(ft,k);
  readln(ft,r);
  readln(ft,x0,y0);
  a:=x0; b:=y0;
  len:=8*arctan(1)*r;
  for i:=1 to k-1 do
  begin
```

```
    readln(ft,x,y);
    len:=len+dlina(x0,y0,x,y);
    x0:=x; y0:=y;
  end;
  len:=len+dlina(x0,y0,a,b);
  writeln(len:10:2);
  close(ft);
End.
```

Задача 6. Где виза?

Медведь решил усилить неприкосновенность своей территории, введя для всяких Волков визовый режим. Как вы помните, Медведь в душе был математик, поэтому для получения визы всякие Волки должны были отгадать кодовое слово по следующему алгоритму: Медведь записывал *K* длинных целых чисел. Волк должен был сложить все цифры каждого числа и получить новые числа. Далее нужно было сложить все цифры в каждом полученном числе и повторять такие действия до тех пор, пока от каждого числа не останется число, меньшее десяти. Для каждого из полученных чисел нужно найти соответствующую букву в алфавите «Все для всяких Волков». Эти буквы и образуют кодовое слово.

Требуется написать программу, которая поможет всяким Волкам отгадать кодовое слово и получить визу для посещения территории Медведя.

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 нескольких строк.

В первой строке содержатся десять букв алфавита «Все для всяких Волков».

Во второй строке содержится единственное натуральное число *K* – количество букв в кодовом слове.

В следующих *K* строках содержатся *K* длинных целых чисел, записанных Медведем. Длина каждого числа не больше 256.

$1 < K < 200$

Формат выходных данных:

Выходной файл *OUTPUT.TXT* содержит кодовое слово, необходимое для получения всяким Волкам визы для посещения территории Медведя.

Пример файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
ВЕЛИЯТФЩДУ 5 12057915491 100579153 200668244	ДЯТЕЛ

100000000009 8111111111111	

Система оценки.

Для оценки правильности решения предоставлено 5 пар файлов INPUT-ANSWER с эталонными входными и выходными данными

Каждый успешно пройденный тест — 20 баллов.

Решение:

Посчитаем сумму цифр в очередном длинном числе, представив его строкой. Будем считать сумму цифр в полученном числе до тех пор, пока она не станет меньше 10. Затем вырежем нужную цифру из алфавита и сформируем кодовое слово.

```
Program task01;
  const n=10;
  type str = string [n];
       chislo = string;
       mas = array[0..n-1] of char;
  var k,i,j,h,x: integer; ft:text;
      alf:str; c:chislo;
      m:mas;

  function sum_cifr(a: integer): integer;
    var sum:integer;
  begin
    sum:=0;
    while a>0 do
    begin
      sum:=sum+a mod 10;
      a:= a div 10;
    end;
    sum_cifr:=sum;
  end;

Begin
  assign(ft,'chisla.txt');
  reset(ft);
  readln(ft,alf);
  readln(ft,k);
  h:=0;
  for i:=1 to k do
  begin
    readln(ft,c);
    x:=0;
    for j:=1 to length(c) do
      x:=x+strtoint(c[j]);
    while x>=10 do
      x:=sum_cifr(x);
    m[h]:=alf[x+1];
    h:=h+1;
  end;
```

```
  for i:=0 to k-1 do
    write(m[i]);
  close(ft);
end.
```

Задача 7. Решишь мою задачку?

В лесной школе лисят учат хитрости и смекалке. И часто для разминки используют числа и некоторые их свойства. Два лисенка узнали о том, что если есть два натуральных числа a и b, то можно играть так, что большее число заменять на разность большего и меньшего, и продолжать игру до тех пор, пока оба числа не станут равными. В этот момент игра заканчивалась.

Их старший брат Хитрый Лис по данным числам и результату, полученному игроками, очень быстро определял, не допустили ли лисята ошибку в процессе игры. И ни разу не ошибся в оценке.

Тогда Медведь, который в душе был математиком, предложил ему решить другую задачу, в которой заданы числа a, b, c, d. Требуется узнать, наступит ли в процессе игры для заданной пары чисел (a, b) такой момент, когда в процессе проверки очередной пары чисел число a будет равно c, а число b будет равно d?

Требуется написать программу, которая поможет решить Хитрому Лису задачу Медведя.

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 одной строки, в которой чрез пробел записаны целые числа a, b, c и d, где $1 \le a, b, c, d \le 10^{18}$

Формат выходных данных:

Выходной файл *OUTPUT.TXT* должен содержать одну строку. Если в процессе применения алгоритма Евклида к паре чисел (a, b) в какой-то момент получается пара (c, d), то строка состоит из слова “YES”, в противном случае – из слова “NO”.

Пример файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
60 150 60 30	YES
60 150 30 60	NO
1000 150 75 50	NO
768 112 80 16	YES

Система оценки.

Для оценки правильности решения предоставлено 20 пар файлов INPUT-ANSWER с эталонными входными и выходными данными

Каждый успешно пройденный тест — 5 баллов.

Решение:

Обратите внимание на то, что трассировка изменений значений переменных *a, b* позволяет разбить процесс вычисления наибольшего общего делителя на фазы, в каждой из которых поочередно остается неизменным значение или *a* или *b*.

Для пятого теста это:

1 фаза: (6012, 2604) = (3408, 2604) =(804, 2604) =

2 фаза: (804, 1800) = (804, 996) =(804, 192) =

3 фаза: (612, 192) =(420, 192) =(228, 192) = (36, 192) =

4 фаза: (36, 156) =(36, 120) =(36, 84) = (36, 48) = (36, 12) =

5 фаза: (24, 12) =(12, 12) =(12, 0)

(6012, 2604) = 12

В соответствии с постановкой задачи нам необходимо убедиться в том, что на одной из итераций, неважно какой, промежуточные значения *a* и *b* совпадут соответственно со значениями *c* и *d*.

Следует обратить внимание на то, что:

- ✓ каждая фаза заканчивается в тот момент, когда меняется истинность отношения «больше-меньше» между значениями *a* и *b;*
- ✓ в каждой фазе значения изменяемого элемента при делении на постоянный элемент дают один и тот же остаток;

При этом пару (*c, d*) ищем:

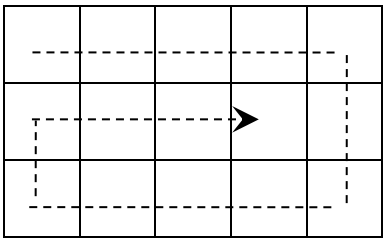
- ✓ среди текущих значений пар (*a, b*) – очень тяжелый алгоритм, и на указанных диапазонах входных данных работает недопустимо медленно;
- ✓ на основании контроля начала и конца фазы, совпадения постоянного элемента фазы с соответствующим значением и анализа совпадений второго компонента пары.

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

bool poisk (long long int a, long long int b,long long int c, long long
int d)
{
    bool rezultat = false;
    while (!(a*b==0)&&!(rezultat)) {// условие можно усилить a>=c и b>=d
        if (a > b) {
            a = a % b;
            if (b == d)
                if ((c - a % b) % b == 0) rezultat = true;
        }
        else {
              b = b % a;
              if (a==c)
```

```
                    if ((d - b % a) % a == 0) rezultat = true;
        }

    }

return rezultat;
}
int main()
{
    long long int chislo1, chislo2, chislo3, chislo4;
    cin >> chislo1;
    cin >> chislo2;
    cin >> chislo3;
    cin >> chislo4;
    cout << poisk(chislo1, chislo2,chislo3, chislo4) << endl;
    system("pause");
    return 0;
}
```

Задача 8. Найду!

Во время очередного обхода леса Медведь, его хозяин, услышал, как Заяц хвастается Ежу, что нашел сокровища, надежно спрятал их в лесу и поставил условные знаки для опознавания. Медведя возмутило самоуправство всяких там Зайцев, и он решил найти все спрятанные сокровища. Поскольку Медведь был в душе математиком, то быстро составил план поиска. Он учел, что важный для него участок леса можно представить в виде прямоугольника с целыми длинами сторон (*n* - высота прямоугольника, *m* - длина прямоугольника), и это позволит разработать стратегию и страховку на случай, чтобы не запутаться. Медведь в единицу времени оценивает наличие сокровищ в квадрате леса размера 1^x1. Перед началом поиска Медведь встал перед левой верхней клеткой прямоугольника в направлении «слева - направо», после чего начал обход, двигаясь по спирали по часовой стрелке. Ну вот примерно, как на рисунке:



n=3
m=5

При этом его тропинка «закручивается» внутрь, захватывая постепенно все новые участки леса. Поиск заканчивается, когда проверены все участки.

Требуется написать программу, которая для заданных величин длины и ширины участка леса определяет количество поворотов, которые должен выполнить Медведь в процессе поиска сокровищ.

Технические требования:

Имя входного файла: *INPUT.TXT*

Имя выходного файла: *OUTPUT.TXT*

Ограничение по времени тестирования: 1 секунда на один тест.

Формат входных данных:

Входной файл *INPUT.TXT* содержит два целых числа, расположенных в одной строке в следующем порядке: n, m ($1 \le n, m \le 32767$). Числа в строке разделены пробелами.

Формат выходных данных:

Выходной файл *OUTPUT.TXT* должен содержать одно целое значение - количество поворотов.

Пример файлов входных и выходных данных:

INPUT.TXT	*OUTPUT.TXT*
3 5	4
4 5	6

Система оценки.

Для оценки правильности решения предоставлено 10 пар файлов INPUT-ANSWER с эталонными входными и выходными данными

Каждый успешно пройденный тест — 10 баллов.

Решение:

В качестве моделей для нахождения алгоритма решения данной задачи могут служить прямоугольники разных видов, и принципиально различными будут те случаи, в которых высота прямоугольника будет больше или меньше, чем длина.

Рассмотрим случай, при котором высота прямоугольника будет меньше. В качестве элемента наблюдения выделим полный контур одного обхода Медведя. Понятно, что все клетки полного контура находятся на одной глубине удаления от краев прямоугольника и, что по завершении обхода контура для продолжения пути Медведь должен сделать поворот для перехода в «исходное состояние». Отметим, что при этом Медведь обходит две вертикальные и две горизонтальные полосы прямоугольника. Длины горизонтальных полос не имеют значения, а вот количество вертикальных определяет количество полных контуров. Из рассуждений выше понятно, что оно равно $n \vdots 2$ и каждый полный контур требует четырех поворотов. При этом важно, является ли число n четным или нет. Если да, то при прохождении самого внутреннего контура Медведь сделает только два поворота из четырех. Если же число полос нечетно, то после прохождения максимально возможного числа полных контуров, Медведь просто пройдет по оставшейся единственной полосе до конца. Ни одного поворота при этом не будет сделано.

Рассмотрим случай, при котором высота прямоугольника будет больше или равна длине. В этом случае количество возможных полных контуров обхода определяется числом m. Если данное число четно, то при прохождении последнего из $m \vdots 2$ контуров отпадет необходимость на последней клетке переходить Медведю в «исходное состояние», так как путь закончен и дальше идти некуда. Если же m

нечетно, то надо из «исходного состояния» сделать шаг «вперед», провернуть «направо» и проделать до конца оставшийся путь.

Понятно, что условия задачи требуют использования целого типа данных для *n* и *m*.

```
program sokrovisha;
   uses crt;
   var n,m,k:integer;
   f,f1:text;
Begin
   clrscr;
   assign(f,'input.txt');
   reset(f);
   read(f,n,m);
   if (n=1) or (m=1) then k:=0
   else
   if n<=m then k:=2*(n-1) else k:=2*m-1;
   assign(f1,'output.txt');
   rewrite(f1);
   writeln(f1,k);
   close(f);
   close(f1);
   End.
```