South African Computer Olympiad
3rd Training Camp 2012
Training Camp Day 1

Overview

Author(s)	**Marco Gallotta**	**Mark Danoher**	**IOI 2007**
Problem	**factory**	**packing**	**flood**
Source	factory.c factory.cpp	packing.c packing.cpp	flood.c flood.cpp
Input file	stdin	stdin	stdin
Output file	stdout	stdout	stdout
Time limit	1 second	1 second	2 seconds
Memory limit	64MiB	64MiB	64MiB
Number of tests	10	10	10
Points per test	10	10	10
Detailed feedback	No	No	No
Total points	**100**	**100**	**100**

The maximum total score is 300 points.

Square Can Factory

Marco Gallotta

Introduction

Fred the manic storekeeper has gone and purchased himself a can factory. He's unhappy with the wastage produced by the usual circular cans and wants to start making square ones. The aluminium sheets from which the tops of the cans are cut are circular and there's little Fred can do to change this.

Task

Fred has some of his worker's frantically against his strategy saying that square cans are less efficient. He's asked for your help in working out the number of square can tops he can cut from a circular sheet.

Some of his machines are broken. When using a broken machine cutting can only start at the center of the sheet as illustrated in Figure 1. When using a fully functional machine, the cutting can begin anywhere within the sheet.

If a square *just* fits in the sheet touching the boundary then it still counts as an additional square top.

Example

A circular sheet of diameter 26 can produce twelve 5×5 square tops when using a broken machine, as illustrated in Figure 1. However, using a fully functional machine allows Fred to adjust the cuts to squeeze in an addtional two cans, giving a total of fourteen tops when using a fully functional machine. Trying to cut any more would result in only partial squares.

Input (stdin)

The first line of input contains a single integer, which is a 1 when using a broken machine or a 0 when using a fully functional machine. The second line contains an integer D, the diameter of the circular sheet. The third line contains an integer S, the size of the square can tops.

Sample input

```
1
26
5
```

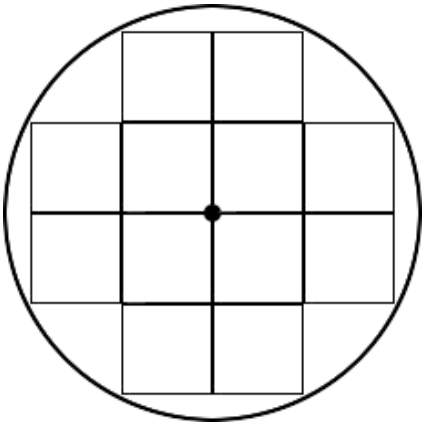


Figure 1: Example sheet from which a maximum of twelve square tops can be cut using a broken machine. The center is marked with a solid circle.

Output (stdout)

The output should contain a single integer, the maximum number of $S \times S$ squares Fred can cut from the sheet.

Sample output

```
12
```

Constraints

- $1 \leq D \leq 50\,000$
- $1 \leq S$
- It will always be possible to cut at least one square from the sheet

Additionally, in 50% of the test cases:

- Fred will always use a broken machine

Time limit

1 second.

Scoring

A correct solution will score 100%, while an incorrect solution will score 0%.

Train Packing

Mark Danoher

Introduction

Fred the manic store-keeper needs to transport some items to a store in another city. He has a train which he will use for this, but he needs your help to determine the best way to pack the items onto the train.

Task

Fred has N items of varying weights that he would like to transport, with the weight of item N_i given as W_i. He has a train with M cars to carry his cargo. All cars have the same capacity C, which is the maximum combined weight of items which can be loaded into each car. Fred would like to keep the items organised so that none are lost along the way. For this reason, he will only allow items to be loaded in to the cars sequentially: items $N_1, \ldots, N_i$ will go in the first car, items $N_{i+1}, \ldots, N_j$ go in the second car, items $N_{j+1}, \ldots, N_k$ in the third car, and so forth. The combined weight of the items loaded into a car cannot exceed the capacity of the car, a car can be left empty (have no items loaded into it), and it is guaranteed that no single item will weigh more than the maximum capacity of a car.

Fred has been advised to balance his load of items across the cars so that they all wear evenly and has been given an algorithm to help him do this. The wasted capacity of a car is defined as the square of the difference between the maximum capacity of the car, C, and the total weight of all items loaded in that car. For a car loaded with items $N_i, \ldots, N_j$ the wasted capacity is then $\left(C - \sum_{x=i}^{j} W_x\right)^2$. The total wasted capacity for the train is the sum of the wasted capacities of each of the cars.

Fred would like you to help him determine the minimum total wasted capacity that can be achieved after packing all of the items onto the train. He is aware that it will sometimes be impossible to fit all of the items onto the train, and he would like you to instead tell him if this is the case.

Example

Suppose that Fred has 3 items he would like to transport and he has a train with 2 cars with a maximum capacity of 4 for the cars. The 3 items have weights of 3, 1 and 2 respectively. Fred could pack the first two items into the first car, which would then be carrying a total weight of 4, and the third item into the second car, which would be carrying a total weight of 2. This would give him a total wasted capacity of $(4-4)^2 + (4-2)^2 = 4$, which is not optimal. The correct solution is to pack the first item into the first car, for a weight of 3, and the next two items into the second car, for a weight of 3 as well. This would give him the minimum total wasted capacity of $(4-3)^2 + (4-3)^2 = 2$.

Input (stdin)

The first line of the input contains three space-separated integers, N, M and C. The following N lines each contain a single integer, the weight W_i of the i^{th} item.

Sample input

```
3 2 4
3
1
2
```

Output (stdout)

Your output should consist of a single integer: the minimum possible total wasted capacity over the entire train. If it is not possible to load all of the items onto the train, you should output -1 instead.

Sample output

```
2
```

Constraints

- $1 \leq N \leq 400$
- $1 \leq M \leq 200$
- $1 \leq C \leq 500$
- $1 \leq W_i \leq C$

Additionally, in 40% of the test cases:

- $1 \leq N \leq 15$
- $1 \leq M \leq 10$

Time limit

1 second.

Scoring

A correct solution will score 100% while an incorrect solution will score 0%.

Flood

IOI 2007

Introduction

A catastrophic flood has struck the city of Cape Town. Many buildings were completely destroyed when the water struck their walls. In this task, you are given a simplified model of the city before the flood and you should determine which of the walls are left intact after the flood.

Task

The model consists of N points in the coordinate plane and W walls. **Each wall connects a pair of points and does not go through any other points.** The model has the following additional properties:

- No two walls intersect or overlap, but they may touch at endpoints;
- Each wall is parallel to either the horizontal or the vertical coordinate axis.

Initially, the entire coordinate plane is dry. At time zero, water instantly floods the exterior (the space not bounded by walls). After exactly one hour, every wall with water on one side and air on the other breaks under the pressure of water. Water then floods the new area not bounded by any standing walls. Now, there may be new walls having water on one side and air on the other. After another hour, these walls also break down and water floods further. This procedure repeats until water has flooded the entire area.

Example

An example of the process on the sample input is shown in Figure 2.

Input (stdin)

The first line of input contains an integer N, the number of points in the plane. Each of the following N lines contains two integers X and Y, the coordinates of one point. The points are numbered 1 to N in the order in which they are given. No two points will be located at the same coordinates.

The following line contains an integer W, the number of walls. Each of the following W lines contains two different integers A and B, meaning that, before the flood, there was a wall connecting points A and B. The walls are numbered 1 to W in the order in which they are given.

Sample input

```
15
1 1
8 1
4 2
7 2
2 3
4 3
6 3
2 5
4 5
6 5
4 6
7 6
1 8
4 8
8 8
17
1 2
2 15
15 14
14 13
13 1
14 11
11 12
12 4
4 3
3 6
6 5
5 8
8 9
9 11
9 10
10 7
7 6
```

Output (stdout)

The first line of output should contain a single integer K, the number of walls left standing after the flood. The following K lines should contain the indices of the walls that are still standing, one wall per line, in order from smallest index to largest.

Sample output

```
4
6
15
16
17
```

Constraints

- $2 \leq N \leq 100\,000$
- $0 \leq X_i, Y_i \leq 1\,000\,000$
- $1 \leq W \leq 2N$
- $1 \leq A_i, B_i \leq N$

Additionally, in 60% of the test cases:

- $2 \leq N \leq 500$

Additionally, in 40% of the test cases:

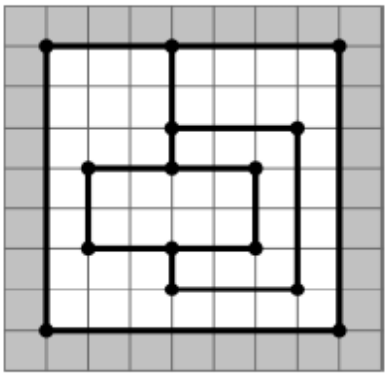
- $2 \leq N \leq 500$
- $0 \leq X_i, Y_i \leq 500$

Time limit

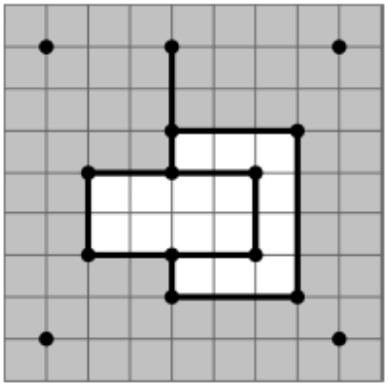
2 seconds.

Scoring

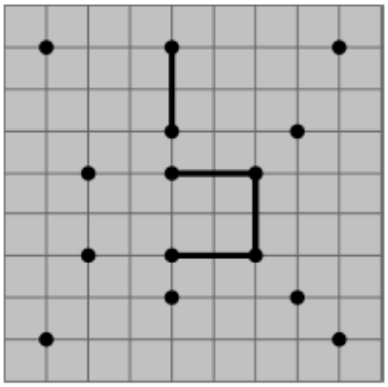
A correct solution will score 100% while an incorrect solution will score 0%.



The state at time zero. Shaded cells represent the flooded area, while white cells represent dry area (air).



The state after one hour.



The state after two hours. Water has flooded the entire area and the 4 remaining walls cannot be broken down.

Figure 2: An example using the sample input data