Overview

Author(s)	Marco Gallotta	Migael Strydom and Marco Gallotta	Max Rabkin
Problem	**contain**	**lasers**	**banquet**
Source	contain.java contain.py contain.c contain.cpp contain.pas	N/A	banquet.java banquet.py banquet.c banquet.cpp banquet.pas
Input file	stdin	lasers.in	stdin
Output file	stdout	lasers.out	stdout
Time limit	1 second	N/A	1 second
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.

Containment

Marco Gallotta

Introduction

Something has gone terribly wrong with the new Large Hadron Collider (LHC) at CERN, and the magnets have started outputting a strange and deadly radiation never seen before. You have to place heavy shielding around them to prevent untold environmental damage.

The LHC is a ring with a control room and magnets placed at various points around the perimeter. The positions of the magnets are represented as the number of meters one must walk anti-clockwise from the control room, around the circumference, to reach the magnet. The ring is R meters in circumference. You have a number of curved shielding blocks, each of which can cover K meters of the ring's circumference. A single shielding block is *not* big enough to cover two magnets that are exactly K meters apart.

Task

Your task is to determine the minimum number of shielding blocks required to cover all magnets. Shielding a magnet with more than one block is acceptable, but every magnet must be shielded.

Example

Consider a ring $R = 10$ meters in circumference, with magnets at 0, 9, 4 and 6 meters. The shielding blocks can each cover 2 meters of the ring's circumference. The magnets at 0 and 9 meters can be covered with a single block, since the ring is circular. The other two magnets require two separate blocks, giving a total of three blocks required to shield the magnets.

Input (stdin)

The first line of input contains three space-separated integers: R, the circumference of the ring; N, the number of magnets; and K, the length (in meters) of the ring's circumference that a shielding block can cover. The next N lines each contain a single integer: the location L_i of a magnet in meters.

Sample input

```
10 4 2
0
9
4
6
```

Output (stdout)

Output a single integer: the minimum number of blocks required to shield all magnets.

Sample output

```
3
```

Constraints

- $1 \leq R \leq 100\,000$
- $0 \leq N \leq R$
- $1 \leq K \leq \min(100, R)$
- $0 \leq L_i < R$ for all i
- $L_i \neq L_j$ for all $i \neq j$

Additionally, in 50% of the test cases:

- $R \leq 1\,000$

Time limit

1 second. Python: 10 seconds.

Scoring

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

Lasers

Migael Strydom and Marco Gallotta

Introduction

Fred the manic storekeeper has finally recruited a team of robots as a security system in one of his big warehouses. Unfortunately, the robots are running a popular operating system that is, alas, broken. The robots moved to random positions in the warehouse and then stopped working — they can no longer move. All is not lost, however, because Fred will equip them with laser scanners to scan and protect the surrounding merchandise in the warehouse.

Task

You are given the (x, y) positions of each of the N robots in the warehouse. You are also given the (x, y) positions of each of the M items of merchandise. Fred wants to equip each robot with a laser. The lasers can scan in a circle around the robot. The strength of a laser is equal to the radius of the circle that it can scan. The cost of giving a robot a laser is equal to the square of its strength, that is, the square of the radius of the circle. Help Fred to equip each robot with a laser of the correct strength so that each item of merchandise is scanned by at least one laser, and so that the total cost of the lasers is minimized. Note that the total cost is the sum of the costs of each robot's laser. You do not have to equip every robot with a laser.

Example

Suppose there are three robots at positions $(1, 1)$, $(5, 3)$ and $(1, 6)$. There are 3 items of merchandise. They are at positions $(2, 1)$, $(7, 3)$ and $(3, 2)$. In this case, the lowest total cost is achieved by equipping the robot at position $(1, 1)$ with a laser of strength 1. It can then scan the item at position $(2, 1)$. The robot at position $(5, 3)$ gets a laser of strength $\sqrt{5}$, and so it can scan the other two items. The final robot gets no laser. The total cost is therefore $1^2 + (\sqrt{5})^2 + 0^2 = 6$, which is indeed the lowest possible total cost.

Input (lasers.in)

The first line of input contains two space-separated integers, N followed by M. The following N lines each contain two space-separated integers, the x and y coordinates of a robot. The next M lines again each contain two space-separated integers, the x and y coordinates of an item of merchandise.

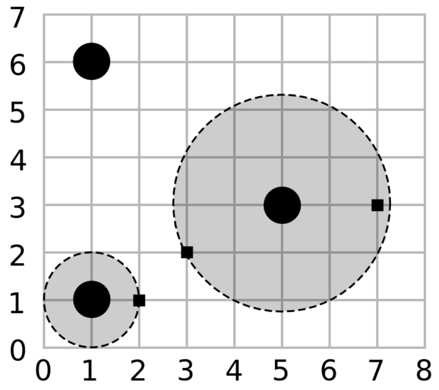

Figure 1: *Diagram of the example. The solid circles represent robots, and the squares represent items of merchandise. The larger shaded circles indicate the regions that the robots' lasers can scan.*

Sample input

```
3 3
1 1
5 3
1 6
2 1
7 3
3 2
```

Output (lasers.out)

Output N lines, each containing a single integer. The integer on line i should be the cost of the laser to give to the robot ranked at position i in the input. If the robot should not be equipped with a laser, output 0 for that line.

Sample output

```
1
5
0
```

Scoring

If the output does not conform to the output specification, you will receive 0 for that test case. If any of the items of merchandise are not scanned by the robots, you will also receive 0. Otherwise, your score will depend on how close your answer is to the best answer found by any of the other competitors.

Let C be the total cost you have found as an answer, and let K be the lowest total cost any contestant has found. Your score for the test case will be the following:

$$\begin{cases} 10 & \text{if } C = K \\ \left\lfloor 8 \times e^{-4\left(\frac{C-K}{K}\right)^2} \right\rfloor & \text{otherwise.} \end{cases}$$

Here $\lfloor x \rfloor$ means the greatest integer not larger than x.

Note that the exponent e^x is calculated as follows:

- In Python: `math.exp(x)`, importing `math` .
- In Java: `Math.exp(x)`, found in `java.lang` .
- In C++: `exp(x)`, in the header `cmath` .

Disgusting Banquet

Max Rabkin

Introduction

As one of the last two surviving members of the human race, you have been invited to the Galactic Conference on Endangered Species. Unfortunately, the conference is being hosted in the Brantisvogon cluster, home to the Vogons: the third worst poets and worst cooks in the universe.

You will manage to survive most of the conference on the packets of peanuts you brought with you from Earth, but you will have to attend the banquet at the end of conference.

You hope to spend as much time as possible hiding in the bathroom during the banquet.

Task

You have been given a menu of N courses, and you have assigned them "tastiness" scores—some of which will be negative.

You can leave the banquet at most K times to go to the bathroom. If you go more often than that, your hosts may become suspicious and force-feed you; however, once you have left, you can stay away as long as you like. You may go to the bathroom before the first course, but it does count as one of your K times.

If you are in the banquet hall for a course, you have to eat it or risk offending your hosts (which is not a risk you can afford to take).

You must write a program to schedule your bathroom visits so that the total tastiness of the food you consume is maximised.

Example

Consider the following menu.

Dish	Tastiness
Dog Liver and Rotten Spinach Soup	−9001
Roast of Jewel-backed Crab	3
Toe Jam Roly-Poly	−364

If $K = 1$, you are best off spending the whole meal in the bathroom, leaving before the first course and coming back after the third. This gives a total tastiness of zero.

On the other hand, if $K = 2$, you can leave before the first course, return before the second and then leave again before the third, for a total tastiness of three.

Input (stdin)

The first line of input contains two integers, N and K. Line $i + 1$ contains a single integer T_i, the tastiness of course i, for i from 1 to N.

Sample input

```
3 2
-9001
3
-364
```

Output (stdout)

Output a single line containing a single integer, the maximum total tastiness.

Sample output

```
3
```

Constraints

- $1 \leq N \leq 50\,000$
- $0 \leq K \leq N$
- $-10\,000 \leq T_i \leq 10\,000$

Additionally, in 60% of the test cases:

- $N \leq 1\,000$

Additionally, in 30% of the test cases:

- $N \leq 15$

Time limit

1 second. Python: 10 seconds.

Scoring

If you find the maximum tastiness for a test run, and format your output correctly, you will score 100%. Otherwise, you will score zero for that test run.