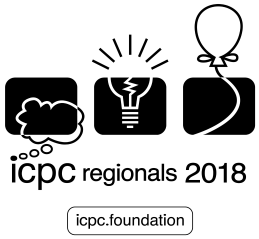
Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter



Central Europe Regional Contest 2018

The ABCD Murderer

`abcd.c`, `abcd.cpp`, `Abcd.java`, `abcd.py`

Oscar enjoys watching crime movies very much. He admires all those villains for how creative some of them can get. He would like to show off some of his creativity, too. Sadly, he's quite inexperienced and he is unable to come up with his own kind of a trick. So he would like to get inspired by some old trick. He always loved watching criminals cutting out the letters from newspapers and putting them together to form a ransom text. However, Oscar ain't a total rip off so he came up with his own variant of this trick. He feels that putting the text together letter by letter is simply too boring and time consuming. So he decided to create his ransom note by cutting out whole words.

Oscar bought several mainstream newspapers, therefore he has practically unlimited source of paper to cut the words from. He can cut out any particular word as many times as he wishes. However, he's still limited by the set of words which appear in the newspapers. The trouble is that some words are just not used in the newspapers at all. To make his job a bit easier, he decided to erase all punctuation and all whitespace characters from the ransom note and ignore the case of characters. He also allows the cut-out words to overlap, as far as the overlapping parts contain the same text. Oscar now wonders how many words at minimum he has to cut out from the newspapers to put together the ransom note.

Input Specification

The first input line contains an integer L ($1 \leq L \leq 3 \cdot 10^5$), the number of words found in the newspapers. The next line contains the text of the ransom note. The text is not empty and consists of lowercase English letters only. It is at most $3 \cdot 10^5$ characters long.

Each of the next L lines contains one word appearing in the newspapers, in lowercase English letters. None of these words is empty or longer than the text of the ransom note. All words appearing in the newspapers are listed in the input at least once. Sum of lengths of all words is not greater than $3 \cdot 10^5$.

Output Specification

Output the minimal number of words Oscar has to cut out from the newspapers to compose his ransom note. If it is not possible to compose the ransom note, print -1. Each word has to be counted as many times as it is physically cut out from a newspaper.

Sample Input 1

```
3
aaaaa
a
aa
aaa
```

Output for Sample Input 1

```
2
```

Sample Input 2

```
5
abecedadabra
abec
ab
ceda
dad
ra
```

Output for Sample Input 2

```
5
```

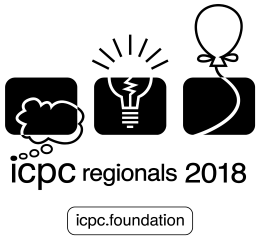
Sample Input 3

```
9
icpcontesticpc
international
collegiate
programming
contest
central
europe
regional
contest
icpc
```

Output for Sample Input 3

```
3
```

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter



Central Europe Regional Contest 2018

The Bridge on the River Kawaii

`bridge.c`, `bridge.cpp`, `Bridge.java`, `bridge.py`

In a very distant land called Midsommer, there is a cute little river delta. Deep purple acid flows in the river, so it is impossible to swim there. There are several islands in the delta and bridges connecting some pairs of them. Every bridge is assigned a *danger level*, which measures how dangerous it is to travel across that bridge, higher levels being more dangerous.

A detective, and as matter of coincidence also a mystery novels author, Richard Hradecki often needs to travel between the islands to pursue his detective cases. Among all possible paths, he prefers to choose the safest one, which means the highest danger level of a bridge on the path must be as low as possible.

For planning purposes, Richard usually asks you to find him the safest path from one island to another island where he is investigating. To be able to satisfy his requests, you have to continuously register three types of events:

- A new bridge between two islands was just built by members of a local tribe.
- A big pink fluffy acid bear Lug appears and destroys a bridge.
- Richard asks you to find the safest path between two islands.

Input Specification

The first line of input contains two integers N and Q ($2 \leq N \leq 10^5$ and $1 \leq Q \leq 10^5$). N is the number of islands (which are labeled $0, 1, \ldots, N-1$) and Q is the number of events to follow.

Each of the next Q lines represents one event and it contains three or four integers, interpreted as follows:

- $0\ X\ Y\ V$: A bridge of *danger level* V ($0 \leq V < 10$) has just been built between islands X and Y.
- $1\ X\ Y$: The bridge connecting islands X and Y has just been destroyed.
- $2\ X\ Y$: Richard asks what is the safest path from island X to island Y.

For all types of events, X and Y denote a valid pair of islands ($0 \leq X, Y < N$ and $X \neq Y$). There is always at most one bridge between any pair of islands. A bridge to be destroyed always exists at that moment.

Output Specification

For each event of type 2, output a line with the *danger level* of the most dangerous bridge on the safest path from X to Y. If there is no path between X and Y, output -1.

Sample Input 1

```
6 15
0 1 2 1
2 1 4
2 1 5
0 2 3 2
2 1 4
2 1 5
0 3 4 3
2 1 4
2 1 5
0 4 5 4
2 1 4
2 1 5
1 4 5
2 1 4
2 1 5
```

Output for Sample Input 1

```
-1
-1
-1
-1
3
-1
3
4
3
-1
```

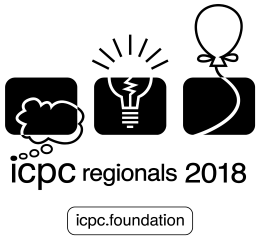
Sample Input 2

```
6 6
0 2 0 4
0 3 4 3
0 0 4 1
0 2 5 4
2 3 2
2 4 2
```

Output for Sample Input 2

```
4
4
```

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter



Central Europe Regional Contest 2018

Clockwork ||ange

`clockwork.c`, `clockwork.cpp`, `Clockwork.java`, `clockwork.py`

Rabbits are small mammals in the family Lepo*r*idae of the *o*rder Lagom*or*pha. So says Wikipedia. C*or*rectly. The c*or*ollary is that they are not b*or*ing, f*or* they are all *o*riginal and well *o*rganized. The rabbits on our farm live in guarded corrals whose borders are adorned with elaborate floral ornaments. Scores of adorable orange carrot clumps grow in the corrals. Rabbits reproduce quickly (it is a norm, hordes of rabbits are born each year) and our mentors are keen to locate them in the corrals effortlessly.

The corrals are well-ordered, they form one straight row of corrals. At the beginning of the first breeding season, some corrals may be unoccupied by rabbits. At the end of each breeding season, a well orchestrated relocation of rabbits is performed. The relocation is guided by a simple formula which depends on one positive integer parameter K, which may be chosen arbitrarily for each season. The relocation works on all corrals in parallel. In each corral, approximately one half of rabbits are removed from the corral and moved K corrals down the row of the corrals. It does not matter whether the target corral is already occupied by some rabbits or not.

If a corral is too close to the end of the row (there are less than K corrals down the row of corrals) then all the rabbits stay in the corral and are not moved anywhere.

Any corral can accommodate unlimited number of rabbits and there are always enough rabbits to breed successfully in any nonempty corral.

You are given a specification of which corrals are occupied and which are empty at the beginning of the first breeding season. Determine the minimum number of relocations needed to populate all corrals by rabbits.

Input Specification

Input consists of a single line with a string of b characters ($1 \leq b \leq 40$), each representing one corral and being either `0` (empty corral) or `1` (inhabited corral). The first character corresponds to the first corral in the row.

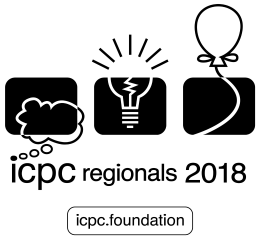
Output Specification

Output the minimum number of relocations needed to populate all corrals. If it is not possible to populate all corrals by any number of relocations, print -1.

Sample Input 1	Output for Sample Input 1
`1010`	`1`

Sample Input 2	Output for Sample Input 2
`100110000`	`3`

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter



Central Europe Regional Contest 2018

Reservoir Dog

`dog.c`, `dog.cpp`, `Dog.java`, `dog.py`

You came to play with your dog to a water reservoir near your house. The dog likes to play with frisbee and since you are bored, you are curious how much time it takes your dog to catch the thrown frisbee and bring it back to you. So you hold your dog by your side and start your stopwatch. A few moments later, in time T_f milliseconds, you throw the frisbee horizontally from height H_f millimeters with velocity V_f millimeters per millisecond. The vertical acceleration due to gravity specific to your frisbee is 1 millimeter per millisecond squared.

In time T_d milliseconds, you release the dog, whose maximal horizontal velocity is V_d. The dog races across ideally flat terrain and it is smart enough to minimize the time needed to bring the frisbee back to you. In order to achieve this, the dog can jump up to height H_d millimeters, and the jump does not influence its horizontal velocity. When the dog catches the frisbee, it immediately races back to you at full speed. You stop the stopwatch when the dog reaches you. Note that you stop the stopwatch even if the dog is in the air, as far as it is directly above the same place as in the beginning.

Your dog is special because it can gain horizontal velocity instantly (there is no speedup or slowdown) and it can change or reverse its horizontal velocity even during a jump while being airborne. Also, the vertical acceleration due to gravity specific to your dog is 3 millimeters per millisecond squared.

For simplicity, assume that both the frisbee and the dog have negligible size. Also, we remind you that a general equation related to the movement of objects in this problem may be expressed in the form $s(t) = s_0 + v_0 t + \frac{1}{2}at^2$. Correct application of the ideas behind this equation is up to you, of course.

Input Specification

Input consists of six space separated integers $T_f, V_f, H_f, T_d, V_d, H_d$. You may assume that each of these integers is between 1 and 10^6 (inclusively) and also that $H_d < H_f$ and $T_f < T_d$.

Output Specification

Output the time measured by your stopwatch in milliseconds, accurate within an absolute error of 10^{-4}.

Sample Input 1

```
1 2 160 20 6 40
```

Output for Sample Input 1

```
31.92569589
```

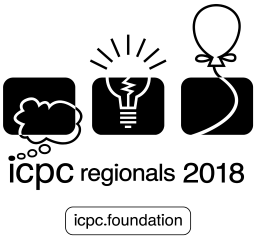
Sample Input 2

```
1 2 160 10 6 40
```

Output for Sample Input 2

```
21.65591118
```

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter



Central Europe Regional Contest 2018

Trees Gump

`gump.c`, `gump.cpp`, `Gump.java`, `gump.py`

The huge trees in the Jumbarian jungle are strategically very important. The Jumbarian army headquarters have selected N trees which are rather close to each other, and decided that a tree house will be built on each of these trees and it will be occupied by a single army unit. Movement of people and material between the tree tops will be supported by the system of bi-directional zip lines connecting pairs of tree tops. For safety reasons, no two zip lines can cross each other (when observed from a satellite).

N units have already been chosen and a list of pairs of these units has been created. Units in a pair are expected to operate on the opposite ends of one zip line. The number of pairs in the list is one less than the number of units. It turns out the list ensures the connectedness of the area, i.e., after the units will be deployed to the trees, each unit will be able to reach any other unit using only zip lines that appear on the list.

All that remains to put this scheme to work is to install the zip lines between tree tops in such a way that will allow to deploy the units in accordance with the above rules.

Input Specification

The first line contains a number N of tree tops and army units ($1 \leq N \leq 1000$). Both the tree tops and the army units are labeled $0, 1, \ldots, N-1$. The next $N-1$ lines contain the list of pairs of units. Each line contains labels of two units expected to operate on the opposite ends of a single zip line. The next N lines give the coordinates of the selected tree tops. The $(i+1)$-th of these lines contains two numbers X_i and Y_i ($0 \leq X_i, Y_i \leq 10^9$), the coordinates of the tree top labeled i. No three tree tops lie on a single line.

Output Specification

Output a list of all pairs of trees which are to be connected by a zip line. The list consists of $N-1$ lines, each line contains two labels of different trees.

If there are multiple solutions to the problem, print any of them.

Sample Input 1

```
5
0 1
1 2
2 3
3 4
0 0
9 9
2 3
3 2
7 8
```

Output for Sample Input 1

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

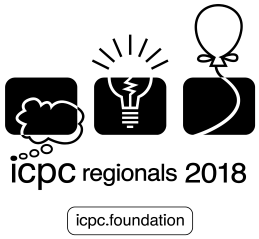
Sample Input 2

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

Output for Sample Input 2

```
0 1
1 2
```

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter



Central Europe Regional Contest 2018

Incredible Hull

`hull.c`, `hull.cpp`, `Hull.java`, `hull.py`

In a royal casino, there is a huge room with many slot machines. The casino manager feels that this room may be too easy to navigate and so the people have a good chance of getting out of the room without losing much money. This needs to be fixed, therefore the manager designed a complicated system of slot machine positions and straight aisles between them, so people get lost there more likely.

The room's shape is roughly oval. If it was empty, the whole room could be observed from any inside point. Each of N slot machines in the room can be represented as a point and it is assigned some unique profit value. The manager wants the slot machines to be connected by aisles in the way described below.

There already are at least 3 machines built into the wall of the room. All these wall machines must be connected by straight aisles to create a closed perimeter path around the whole room. All of the remaining slot machines in the room (if there are any) will be inside the area enclosed by the perimeter path.

The room is to be divided by straight aisles into smaller separate areas, each enclosed by its own perimeter path. The division proceeds in two phases.

- The first phase is ruled by the following scheme. If there are at least 4 slot machines on a perimeter path, two pivot machines have to be chosen. The first pivot machine is the most profitable one on the perimeter path. The second pivot machine is also on the perimeter path and it is the furthest one from the first one. The distance between machines is measured along the perimeter path and it is equal to the number of aisles one has to traverse to get from one machine to the other one. If there are two furthest machines, the more profitable one has to be selected as the second pivot machine. The area inside the perimeter path is divided into two areas by a new aisle which connects the pivot machines. The perimeter path of each of the two new smaller areas is the minimal part of the original perimeter path which together with the new aisle form a closed path encircling the smaller area. The scheme is applied repeatedly until no new area can be created.
- When the previous division phase is completed, the second division phase begins. It is ruled by the following scheme. For any area A with no aisle inside it, consider all slot machines inside that area. If there are any, connect the most profitable one of them by aisles to all slot machines on the perimeter path P_A of the area A. This divides the area into several smaller areas. The perimeter path of each new area consists of one aisle of P_A and two newly created aisles. The area lies inside the triangle formed by these three aisles and it contains no other aisles. The scheme is applied repeatedly until no new area can be created.

The Manager wants to assess the profitability of his design. For this purpose, he defines a so-called *highly-profitable configuration*, which is a maximum-size set of machines with a property that each pair of machines in the set is directly connected by an aisle.

The manager is specifically interested in three *profitability parameters*. The first profitability parameter is the size of one highly-profitable configuration. The second profitability parameter is the number of highly-profitable configurations that exist in the design. The third profitability parameter is the number of slot machines which are part of at least one highly-profitable configuration.

Input Specification

The first line of input contains a single integer N ($3 \leq N \leq 10^5$), the number of slot machines. Each of the next N lines contains two integers X, Y ($0 \leq X, Y \leq 10^9$), denoting the position of one slot machine in the room. The machines are listed in decreasing order of their profit value. The positions of no three machines are collinear.

Output Specification

Output the three profitability parameters.

Sample Input 1

```
6
8 2
6 8
4 9
3 5
3 0
1 5
```

Output for Sample Input 1

```
4 1 4
```

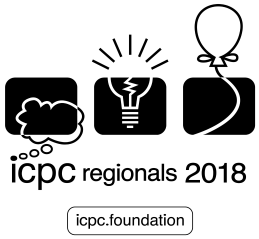
Sample Input 2

```
6
1 1
6 1
1 6
6 6
3 2
4 5
```

Output for Sample Input 2

```
4 2 6
```

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter



Central Europe Regional Contest 2018

Shooter Island

`island.c`, `island.cpp`, `Island.java`, `island.py`

You were recently promoted to captain and your troopers are now operating on a special mission in a storm. The battlefield is kind of extraordinary, it is located above the Arctic Circle on a huge ice floe. You coordinate the actions from the headquarters. There are many high-end computers keeping you up to date on the developments at the battlefield, which is modeled by the AI interface as a grid of unit squares. Each unit square is identified by its row and column index in the grid. Bigger rectangles, consisting of unit squares, are described by a pair of unit squares in the opposite corners of the rectangle. At the beginning, all squares are covered with ice.

There are two important types of information you receive from the computers:

0. Information about hits: Your enemy hit a rectangle described by unit squares $[x_1,y_1]$ and $[x_2,y_2]$. This rectangle is then immediately flooded by cold arctic water.
1. A query by your troopers: They ask whether it is possible to go from square $[x_1,y_1]$ to $[x_2,y_2]$ by a boat. The boat may be represented by a circle of radius 0.31416. Note that the boat has to stay fully on water surface all the time and it is not allowed to leave the battlefield area.

Your troopers need your help! Can you guide them reliably?

Input Specification

The first input line contains an integer L ($1 \leq L \leq 2 \cdot 10^5$), the number of lines to follow. Each of the next L lines contains five integers t, x_1, y_1, x_2, y_2 ($t \in \{0, 1\}$, $1 \leq x_1, x_2 \leq 50$, $1 \leq y_1, y_2 \leq 10^5$). Number t is the type of information and pairs $[x_1, y_1]$ and $[x_2, y_2]$ specify the respective unit squares.

Output Specification

For each query, output 1 if it is possible to sail from unit square $[x_1,y_1]$ to unit square $[x_2,y_2]$, and 0 otherwise.

Sample Input 1

```
6
0 4 4 6 6
0 6 6 7 8
0 1 3 3 3
1 1 7 6 1
1 5 4 6 8
1 4 5 1 3
```

Output for Sample Input 1

```
0
1
0
```

Sample Input 2

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

Output for Sample Input 2

```
1
```

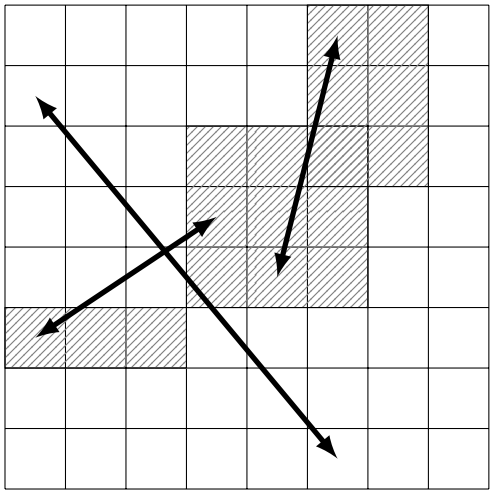
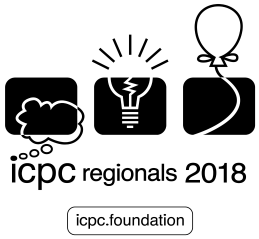


Figure 1: Illustration of Sample Input 1.

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter



Central Europe Regional Contest 2018

The Lord of the Kings

`king.c`, `king.cpp`, `King.java`, `king.py`

After many long and exhausting years of war with dangerous overseas nations, our country has finally managed to get rid of almost all opposing forces and emerged victorious. Such a glorious victory shall be remembered and celebrated for many upcoming years. Because of that, our king has decided to declare the day of the victory as a public holiday, when a victory parade is to be held. During the parade, the king accompanied by his troops will start from his palace and subsequently visit all cities in the country.

The king and his suite will travel by a new type of environment-friendly electric helicopter that has a disadvantage of a relatively short operating radius. The king has requested you and your advisors to build helipads on some farm fields and in all the cities, so that each city is reachable from his palace by a sequence of short flights between helipads. However, building the helipads and the supporting infrastructure is costly. It is thus important to minimize the number of farm fields on which helipad construction takes place.

In addition, due to specific helicopter design, the king and his troops need to move in a special pattern which may influence the number and the locations of helipads.

You are given a rectangular grid map of the country, which consists of tiles of farm fields, tiles of cities and a tile of the king's palace. Also, you are given the movement pattern of the helicopter — it may move either as a Rook, a Queen, a Bishop, a Knight, or as a King in the game of chess (see images for movement illustration). Your task is to determine the minimal number of farm fields tiles and city tiles which have to host a helipad, in order to fulfill the conditions specified above. The tile with the king's palace already has a helipad and does not need a new one.

Input Specification

The first line of input contains two integers N and M ($1 \leq N, M \leq 15$), where N is the number of rows and M is the number of columns in the grid representing our country. The second line of input contains two integers X and Y ($1 \leq X \leq N$, $1 \leq Y \leq M$), representing the position of the king's palace, followed by a single character determining the movement pattern ("`R`" – Rook, "`Q`" – Queen, "`B`" – Bishop, "`N`" – Knight, "`K`" – King). The third line of input contains an integer T ($1 \leq T \leq 10$) representing the number of cities in the country. After that, T lines follow, each containing two integers W and Z ($1 \leq W \leq N$, $1 \leq Z \leq M$). Each such line represents a position of one city tile. All cities occupy different tiles, no city occupies the palace tile. All tiles that do not represent a city or palace are considered to be farm fields.

Output Specification

Output a single integer — the minimal number of farm/city tiles to host a helipad. Should it be impossible for the king and his troops to visit all cities, output -1.

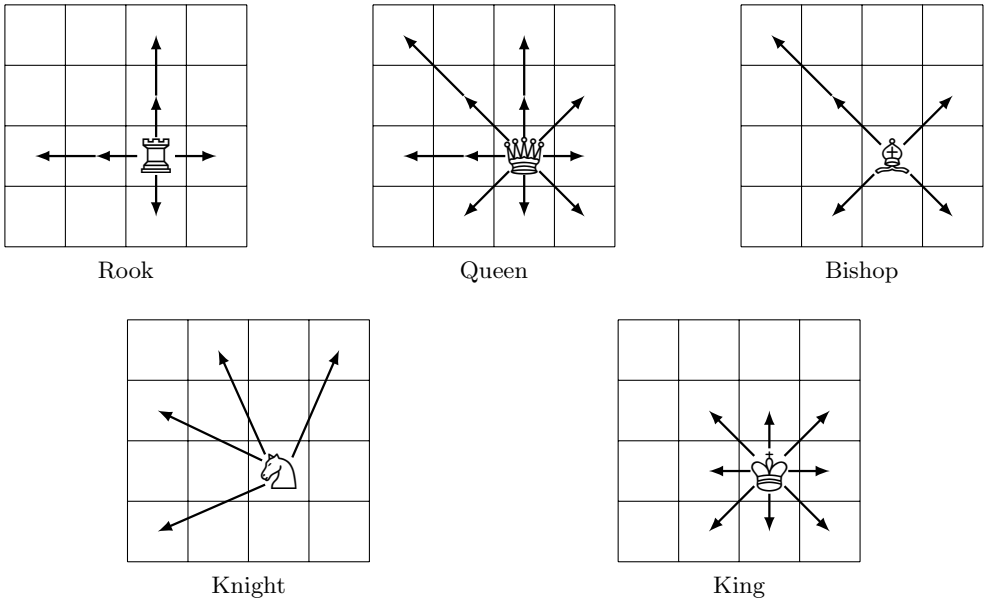


Figure 1: Illustration of specific helicopter movement patterns.

Sample Input 1

```
3 3
3 1 K
2
1 1
1 3
```

Output for Sample Input 1

```
3
```

Sample Input 2

```
3 3
3 1 Q
2
1 1
1 3
```

Output for Sample Input 2

```
2
```

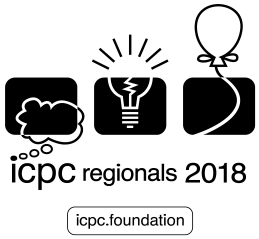
Sample Input 3

```
5 5
4 4 R
4
1 2
2 1
2 5
5 1
```

Output for Sample Input 3

```
6
```

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter



Central Europe Regional Contest 2018

The Silence of the Lamps

`lamps.c`, `lamps.cpp`, `Lamps.java`, `lamps.py`

First, for those who have never seen a lamp before, let's say it is a cuboid (box with rectangular faces) made of glass and filled with gas. All sides of a lamp have integer lengths.

Once upon a time, our lecturer was sentenced for destroying lamps on a street. He must have gone somewhat crazy, as he thought some of the lamps were screaming at him in high-pitched voices.

In his beautiful mind, he followed a weird pattern. He only recognized and destroyed those lamps which had no square face and whose volume did not exceed a fixed value. Later, during a session with his doctor Clarice, he said he was very scared of large objects and of objects with too regular shapes.

Your task is to count all possible shapes matching lecturer's conditions.

Input Specification

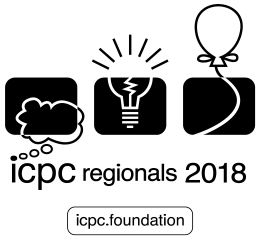
The first input line contains a number T of test cases ($1 \leq T \leq 10^5$). Each of the next T lines contains a single integer N ($1 \leq N \leq 10^6$), the maximum recognizable volume of a lamp.

Output Specification

For each test case output the number of different lamp shapes which could have been destroyed in the rage.

Sample Input 1	Output for Sample Input 1
5	0
5	1
6	3
10	26
30	2406
666	

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter



Central Europe Regional Contest 2018

Matrice

`matrice.c`, `matrice.cpp`, `Matrice.java`, `matrice.py`

Agent Sue Thomas and her son are looking for trinities in a grid. The word *trinity* is a neologism referring to a particular triangular (as the morpheme "tri" suggests) shape composed of cells in the grid.

Each trinity is a result of taking a square-shaped area of the cells and removing all cells that lie either above or below one of the two diagonals of the area. The diagonal may be either the main diagonal (southeast-northwest direction) or the main antidiagonal (southwest-northeast direction). A valid trinity consists of at least three grid cells and all its cells contain the same character.

Input Specification

The first input line contains two numbers N and M ($1 \leq N, M \leq 1000$), describing the number of rows and columns in the grid, respectively. Each of next N lines contains M characters, whose ASCII codes are between 33 and 126, inclusively.

Output Specification

Output the number of different valid trinities in the input grid.

Sample Input 1

```
2 2
AA
Ad
```

Output for Sample Input 1

```
1
```

Sample Input 2

```
5 5
#####
####.
###..
##...
#....
```

Output for Sample Input 2

```
60
```

Sample Input 3

```
5 4
hwwr
eahe
lroy
lswo
oaau
```

Output for Sample Input 3

```
0
```

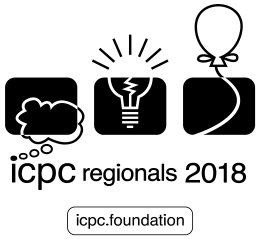
Sample Input 4

```
5 6
#girls
##areb
#.#est
#..#!!
#####!
```

Output for Sample Input 4

```
7
```

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter



Central Europe Regional Contest 2018

Mirrority Report

`report.c`, `report.cpp`, `Report.java`, `report.py`

In the physics class, each student has to complete a semestral work. You chose to work on an experiment which examines the properties of rare unstable particles, known as *red balls*. A red ball particle travels in a straight line until it is reflected by a mirror.

Our experiment can be thought of as several one-reflection mirrors, which can be treated as (potentially infinite) lines in a 2D plane. A one-reflection mirror is made in a way that it reflects a particle only for the first time the mirror is hit by the particle, no matter which direction the particle is coming from. After that, the mirror will always let that particle pass through.

When reflecting, the angle of the line along which the particle approaches the point of reflection is equal to the angle of the line along which the particle leaves the point of reflection, when both angles are measured to the normal of the reflecting mirror. The particle disintegrates if it gets to a crossing point of two mirrors which would otherwise reflect it.

The mirror layout was created by your teacher and your task is to find directions which cause a particle launched from its given starting location to reach a specified final location.

Input Specification

The first input line contains a nonnegative integer N ($0 \leq N \leq 8$), the number of mirrors. The second input line contains four integers S_x, S_y, E_x, E_y, the coordinates of the starting location and the final location. Each of the next N lines contains four integers A_x, A_y, B_x, B_y, describing two points which define the line of the i-th mirror.

All input coordinates are between -100 and 100, inclusively. The starting and the final locations are different. The distance of both starting and final location from any mirror is nonzero.

It is guaranteed that the input does not require a valid solution to reflect the particle closer than 10^{-4} from any crossing point of the reflecting mirror with any other mirror that could still reflect the particle.

Output Specification

Output one line with one integer, the number of distinct launch directions from the starting location which make the particle reach the final location.

Sample Input 1

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

Output for Sample Input 1

```
1
```

Sample Input 2

```
3
2 3 5 2
1 3 1 1
2 4 4 2
5 4 6 5
```

Output for Sample Input 2

```
1
```

Sample Input 3

```
1
0 0 4 0
0 4 4 4
```

Output for Sample Input 3

```
2
```

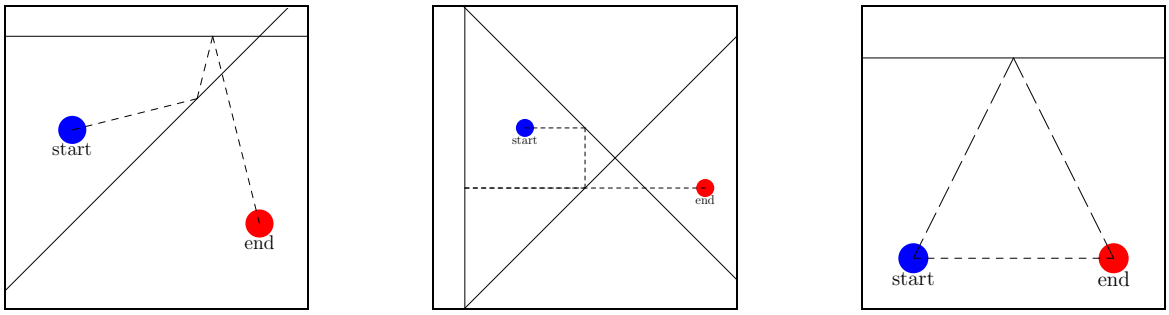
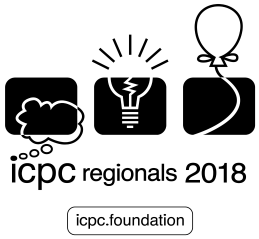


Figure 1: Illustration of Sample Inputs 1, 2, and 3.

Czech Technical University in Prague
ICPC Foundation
Czech ACM Chapter



Central Europe Regional Contest 2018

Game of Stones

`stones.c`, `stones.cpp`, `Stones.java`, `stones.py`

Two players, Petyr and Varys, play a game in which players remove stones from N piles in alternating turns. Petyr, in his turn, can remove at most A stones from any pile. Varys, in his turn, can remove at most B stones from any pile. Each player has to remove at least one stone in his turn. The player who removes the last stone wins.

The game has already started and it is Petyr's turn now. Your task is to determine whether he can win the game if both he and Varys play the game in the best possible way.

Input Specification

The first input line contains three integers N, A, and B ($1 \leq N \leq 10^5$ and $1 \leq A, B \leq 10^5$). N describes the number of piles and A, B represent Petyr's and Varys' restrictions. The second line contains N integers $X_1, \ldots, X_N$ ($1 \leq X_i \leq 10^6$) specifying the current number of stones in all piles.

Output Specification

Output the name of the winner.

Sample Input 1

```
2 3 4
2 3
```

Output for Sample Input 1

```
Petyr
```

Sample Input 2

```
7 8 9
1 2 3 4 5 6 7
```

Output for Sample Input 2

```
Varys
```