Educational Codeforces Round 1

A. Tricky Sum

1 second, 256 megabytes

In this problem you are to calculate the sum of all integers from 1 to n, but you should take all powers of two with minus in the sum.

For example, for $n = 4$ the sum is equal to $-1 - 2 + 3 - 4 = -4$, because 1, 2 and 4 are 2^0, 2^1 and 2^2 respectively.

Calculate the answer for t values of n.

Input

The first line of the input contains a single integer t ($1 \le t \le 100$) — the number of values of n to be processed.

Each of next t lines contains a single integer n ($1 \le n \le 10^9$).

Output

Print the requested sum for each of t integers n given in the input.

input
```
2
4
1000000000
```
output
```
-4
499999998352516354
```

The answer for the first sample is explained in the statement.

B. Queries on a String

2 seconds, 256 megabytes

You are given a string s and should process m queries. Each query is described by two 1-based indices l_i, r_i and integer k_i. It means that you should cyclically shift the substring $s[l_i... r_i]$ k_i times. The queries should be processed one after another in the order they are given.

One operation of a cyclic shift (rotation) is equivalent to moving the last character to the position of the first character and shifting all other characters one position to the right.

For example, if the string s is `abacaba` and the query is $l_1 = 3, r_1 = 6, k_1 = 1$ then the answer is `abbacaa`. If after that we would process the query $l_2 = 1, r_2 = 4, k_2 = 2$ then we would get the string `baabcaa`.

Input

The first line of the input contains the string s ($1 \le |s| \le 10\,000$) in its initial state, where $|s|$ stands for the length of s. It contains only lowercase English letters.

Second line contains a single integer m ($1 \le m \le 300$) — the number of queries.

The i-th of the next m lines contains three integers l_i, r_i and k_i ($1 \le l_i \le r_i \le |s|, 1 \le k_i \le 1\,000\,000$) — the description of the i-th query.

Output

Print the resulting string s after processing all m queries.

input
```
abacaba
2
3 6 1
1 4 2
```
output
```
baabcaa
```

The sample is described in problem statement.

C. Nearest vectors

2 seconds, 256 megabytes

You are given the set of vectors on the plane, each of them starting at the origin. Your task is to find a pair of vectors with the minimal non-oriented angle between them.

Non-oriented angle is non-negative value, minimal between clockwise and counterclockwise direction angles. Non-oriented angle is always between 0 and π. For example, opposite directions vectors have angle equals to π.

Input

First line of the input contains a single integer n $(2 \le n \le 100\,000)$ — the number of vectors.

The i-th of the following n lines contains two integers x_i and y_i $(|x|, |y| \le 10\,000, x^2 + y^2 > 0)$ — the coordinates of the i-th vector. Vectors are numbered from 1 to n in order of appearing in the input. It is guaranteed that no two vectors in the input share the same direction (but they still can have opposite directions).

Output

Print two integer numbers a and b $(a \ne b)$ — a pair of indices of vectors with the minimal non-oriented angle. You can print the numbers in any order. If there are many possible answers, print any.

input

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

output

```
3 4
```

input

```
6
-1 0
0 -1
1 0
1 1
-4 -5
-4 -6
```

output

```
6 5
```

D. Igor In the Museum

1 second, 256 megabytes

Igor is in the museum and he wants to see as many pictures as possible.

Museum can be represented as a rectangular field of $n \times m$ cells. Each cell is either empty or impassable. Empty cells are marked with '.', impassable cells are marked with '*'. Every two adjacent cells of different types (one empty and one impassable) are divided by a wall containing one picture.

At the beginning Igor is in some empty cell. At every moment he can move to any empty cell that share a side with the current one.

For several starting positions you should calculate the maximum number of pictures that Igor can see. Igor is able to see the picture only if he is in the cell adjacent to the wall with this picture. Igor have a lot of time, so he will examine every picture he can see.

Input

First line of the input contains three integers n, m and k $(3 \le n, m \le 1000, 1 \le k \le min(n \cdot m, 100\,000))$ — the museum dimensions and the number of starting positions to process.

Each of the next n lines contains m symbols '.', '*' — the description of the museum. It is guaranteed that all border cells are impassable, so Igor can't go out from the museum.

Each of the last k lines contains two integers x and y $(1 \le x \le n, 1 \le y \le m)$ — the row and the column of one of Igor's starting positions respectively. Rows are numbered from top to bottom, columns — from left to right. It is guaranteed that all starting positions are empty cells.

Output

Print k integers — the maximum number of pictures, that Igor can see if he starts in corresponding position.

input

```
5 6 3
******
*..*.*
******
*....*
******
2 2
2 5
4 3
```

output

```
6
4
10
```

input

```
4 4 1
****
*..*
*.**
****
3 2
```

output

```
8
```

E. Chocolate Bar

2 seconds, 256 megabytes

You have a rectangular chocolate bar consisting of $n \times m$ single squares. You want to eat **exactly** k squares, so you may need to break the chocolate bar.

In one move you can break any single rectangular piece of chocolate in two rectangular pieces. You can break only by lines between squares: horizontally or vertically. The cost of breaking is equal to square of the break length.

For example, if you have a chocolate bar consisting of 2×3 unit squares then you can break it horizontally and get two 1×3 pieces (the cost of such breaking is $3^2 = 9$), or you can break it vertically in two ways and get two pieces: 2×1 and 2×2 (the cost of such breaking is $2^2 = 4$).

For several given values n, m and k find the minimum total cost of breaking. You can eat exactly k squares of chocolate if after all operations of breaking there is a set of rectangular pieces of chocolate with the total size equal to k squares. The remaining $n \cdot m$ - k squares are not necessarily form a single rectangular piece.

Input

The first line of the input contains a single integer t $(1 \le t \le 40910)$ — the number of values n, m and k to process.

Each of the next t lines contains three integers n, m and k $(1 \le n, m \le 30, 1 \le k \le min(n \cdot m, 50))$ — the dimensions of the chocolate bar and the number of squares you want to eat respectively.

Output

For each n, m and k print the minimum total cost needed to break the chocolate bar, in order to make it possible to eat exactly k squares.

input

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

output

```
5
5
4
0
```

In the first query of the sample one needs to perform two breaks:

- to split 2×2 bar into two pieces of 2×1 (cost is $2^2 = 4$),
- to split the resulting 2×1 into two 1×1 pieces (cost is $1^2 = 1$).

In the second query of the sample one wants to eat 3 unit squares. One can use exactly the same strategy as in the first query of the sample.

F. Cut Length

0.5 seconds, 256 megabytes

Given simple (without self-intersections) n-gon. It is not necessary convex. Also you are given m lines. For each line find the length of common part of the line and the n-gon.

The boundary of n-gon belongs to polygon. It is possible that n-gon contains 180-degree angles.

Input

The first line contains integers n and m $(3 \le n \le 1000; 1 \le m \le 100)$. The following n lines contain coordinates of polygon vertices (in clockwise or counterclockwise direction). All vertices are distinct.

The following m lines contain line descriptions. Each of them contains two distict points of a line by their coordinates.

All given in the input coordinates are real numbers, given with at most two digits after decimal point. They do not exceed 10^5 by absolute values.

Output

Print m lines, the i-th line should contain the length of common part of the given n-gon and the i-th line. The answer will be considered correct if the absolute or relative error doesn't exceed 10^{-6}.

input

```
4 3
0 0
1 0
1 1
0 1
0 0 1 1
0 0 0 1
0 0 1 -1
```

output

```
1.41421356237309514547
1.00000000000000000000
0.00000000000000000000
```