A - Daily Cookie

Time Limit: 2 sec / Memory Limit: 1024 MiB

Score: 100 points

Problem Statement

There are N boxes arranged in a row, and some of these boxes contain cookies.

The state of these boxes is represented by a string S of length N. Specifically, the i-th box $(1 \leq i \leq N)$ from the left contains one cookie if the i-th character of S is `@`, and is empty if it is `.`.

Over the next D days, Takahashi will choose and eat one cookie per day from among the cookies in these boxes.

Determine how many of the N boxes will be empty after D days have passed. (It can be proved that this value does not depend on which cookies Takahashi chooses each day.)

It is guaranteed that S contains at least D occurrences of `@`.

Constraints

- $1 \leq D \leq N \leq 100$
- N and D are integers.
- S is a string of length N consisting of `@` and `.`.
- S contains at least D occurrences of `@`.

Input

The input is given from Standard Input in the following format:

$$
\begin{array}{l} N \quad D \\ S \end{array}
$$

Output

Print the number of boxes that will be empty after D days have passed among the N boxes.

Sample Input 1

```
5 2
.@@.@
```

Sample Output 1

```
4
```

For example, Takahashi might act as follows:

- Day 1: There are cookies in the 2nd, 3rd, and 5th boxes from the left. He chooses the cookie in the 2nd box to eat.
- Day 2: There are cookies in the 3rd and 5th boxes. He chooses the cookie in the 5th box to eat.
- After two days have passed, only the 3rd box from the left contains a cookie. Therefore, four out of the five boxes are empty.

Even though Takahashi might choose differently on each day than in this example, there will still be four empty boxes after two days. Therefore, the answer is 4.

Sample Input 2

```
3 3
@@@
```

Sample Output 2

```
3
```

Sample Input 3

```
10 4
@@@.@@.@@.
```

Sample Output 3

```
7
```

B - Daily Cookie 2

Time Limit: 2 sec / Memory Limit: 1024 MiB

Score: 200 points

Problem Statement

This problem shares a similar setting with Problem A. The way Takahashi chooses cookies and what you are required to find are different from Problem A.

There are N boxes arranged in a row, and some of these boxes contain cookies.

The state of these boxes is represented by a string S of length N. Specifically, the i-th box $(1 \leq i \leq N)$ from the left contains one cookie if the i-th character of S is `@`, and is empty if it is `.`.

Over the next D days, Takahashi will choose and eat one cookie per day from among the cookies in these boxes. On each day, he chooses the cookie in the rightmost box that contains a cookie at that point.

Determine, for each of the N boxes, whether it will contain a cookie after D days have passed.

It is guaranteed that S contains at least D occurrences of `@`.

Constraints

- $1 \leq D \leq N \leq 100$
- N and D are integers.
- S is a string of length N consisting of `@` and `.`.
- S contains at least D occurrences of `@`.

Input

The input is given from Standard Input in the following format:

```
N D
S
```

Output

Print a string of length N. The i-th character $(1 \leq i \leq N)$ of the string should be `@` if the i-th box from the left contains a cookie after D days have passed, and `.` otherwise.

Sample Input 1

```
5 2
.@@.@
```

Sample Output 1

```
.@...
```

Takahashi acts as follows:

- Day 1: There are cookies in the 2nd, 3rd, and 5th boxes from the left. Among these, the rightmost is the 5th box. He eats the cookie in this box.
- Day 2: There are cookies in the 2nd and 3rd boxes. Among these, the rightmost is the 3rd box. He eats the cookie in this box.
- After two days have passed, only the 2nd box from the left contains a cookie.

Therefore, the correct output is `.@...`.

Sample Input 2

```
3 3
@@@
```

Sample Output 2

```
...
```

Sample Input 3

```
10 4
@@@.@@.@@.
```

Sample Output 3

```
@@@.......
```

C - Kaiten Sushi

Time Limit: 2 sec / Memory Limit: 1024 MiB

Score: 350 points

Problem Statement

There are N people numbered from 1 to N visiting a conveyor belt sushi restaurant. The **gourmet level** of person i is A_i.

Now, M pieces of sushi will be placed on the conveyor belt. The **deliciousness** of the j-th sushi is B_j. Each piece of sushi passes in front of people $1, 2, \dots, N$ in this order. Each person, when a sushi whose deliciousness is not less than their gourmet level passes in front of them, will take and eat that sushi; otherwise, they do nothing. A sushi that person i takes and eats will no longer pass in front of person j $(j > i)$.

For each of the M pieces of sushi, determine who eats that sushi, or if nobody eats it.

Constraints

- $1 \leq N, M \leq 2 \times 10^5$
- $1 \leq A_i, B_i \leq 2 \times 10^5$
- All input values are integers.

Input

The input is given from Standard Input in the following format:

$$\begin{array}{llll} N & M & & \\ A_1 & A_2 & \dots & A_N \\ B_1 & B_2 & \dots & B_M \end{array}$$

Output

Print M lines. The j-th line ($1 \leq j \leq M$) should contain the number representing the person who eats the j-th sushi, or `-1` if nobody eats it.

Sample Input 1

```
3 3
3 8 2
5 2 1
```

Sample Output 1

```
1
3
-1
```

- For the 1st sushi:
  - It first passes in front of person 1. Since $B_1 \geq A_1$, person 1 takes and eats it.
  - It will not pass in front of person 2 and 3.
- For the 2nd sushi:
  - It first passes in front of person 1. Since $B_2 < A_1$, person 1 does nothing.
  - Next, it passes in front of person 2. Since $B_2 < A_2$, person 2 does nothing.
  - Finally, it passes in front of person 3. Since $B_2 \geq A_3$, person 3 takes and eats it.
- For the 3rd sushi:
  - It first passes in front of person 1. Since $B_3 < A_1$, person 1 does nothing.
  - Next, it passes in front of person 2. Since $B_3 < A_2$, person 2 does nothing.
  - Finally, it passes in front of person 3. Since $B_3 < A_3$, person 3 does nothing.
  - Therefore, nobody eats this sushi.

Sample Input 2

```
3 3
1 1 1
1 1 1
```

Sample Output 2

```
1
1
1
```

Sample Input 3

```
10 5
60 83 76 45 70 91 37 58 94 22
70 39 52 33 18
```

Sample Output 3

```
1
7
4
10
-1
```

D - Keep Distance

Time Limit: 5 sec / Memory Limit: 1024 MiB

Score: 425 points

Problem Statement

You are given integers N and M.

Print all integer sequences $(A_1, A_2, \ldots, A_N)$ of length N that satisfy all of the following conditions, in lexicographical order.

- $1 \leq A_i$
- $A_{i-1} + 10 \leq A_i$ for each integer i from 2 through N
- $A_N \leq M$

▶ What is lexicographical order?

Constraints

- $2 \leq N \leq 12$
- $10N - 9 \leq M \leq 10N$
- All input values are integers.

Input

The input is given from Standard Input in the following format:

```
N M
```

Output

Let X be the number of integer sequences that satisfy the conditions, and print $X + 1$ lines.

The first line should contain the value of X.

The $(i + 1)$-th line ($1 \leq i \leq X$) should contain the i-th smallest integer sequence in lexicographical order, with elements separated by spaces.

Sample Input 1

```
3 23
```

Sample Output 1

```
10
1 11 21
1 11 22
1 11 23
1 12 22
1 12 23
1 13 23
2 12 22
2 12 23
2 13 23
3 13 23
```

$(1, 11, 21), (1, 11, 22), (1, 11, 23), (1, 12, 22), (1, 12, 23), (1, 13, 23), (2, 12, 22), (2, 12, 23), (2, 13, 23), (3, 13, 23)$ are the 10 sequences that satisfy the conditions.

E - Expansion Packs

Time Limit: 2 sec / Memory Limit: 1024 MiB

Score: 475 points

Problem Statement

There are infinitely many packs, each containing N cards. In each pack, the i-th card is rare with probability P_i percent. Whether each card is rare is independent of other cards being rare.

You will now open the packs one by one, and obtain all the cards in each opened pack. When you keep opening packs until you have obtained a total of at least X rare cards, find the expected number of packs you will open.

Constraints

- $1 \leq N \leq 5000$
- $1 \leq X \leq 5000$
- $1 \leq P_i \leq 100$
- All input values are integers.

Input

The input is given from Standard Input in the following format:

$$\begin{array}{l} N \quad X \\ P_1 \quad P_2 \quad \ldots \quad P_N \end{array}$$

Output

Print the answer.

Your answer will be considered correct if the absolute or relative error from the true answer is at most 10^{-6}.

Sample Input 1

```
2 2
50 100
```

Sample Output 1

```
1.5000000000000000
```

The number of packs opened will be 1 with probability $\frac{1}{2}$, and 2 with probability $\frac{1}{2}$.

Sample Input 2

```
2 3
40 60
```

Sample Output 2

```
3.2475579530543811
```

Sample Input 3

```
6 3
10 33 33 10 100 10
```

Sample Output 3

```
1.8657859189536100
```

F - Falling Bars

Time Limit: 2 sec / Memory Limit: 1024 MiB

Score: 525 points

Problem Statement

There is a grid with H rows and W columns. Let (i, j) denote the cell at the i-th row from the top and the j-th column from the left.

There are N horizontal bars numbered from 1 to N placed on the grid. Bar i consists of L_i blocks of size 1×1 connected horizontally, and its leftmost block is initially at cell (R_i, C_i). That is, initially, bar i occupies the cells $(R_i, C_i), (R_i, C_i + 1), \dots, (R_i, C_i + L_i - 1)$. It is guaranteed that there is no cell occupied by two different bars.

The current time is $t = 0$. At every time $t = 0.5 + n$ for some non-negative integer n, the following occurs in order of $i = 1, 2, \dots, N$:

- If bar i is not on the bottom row (the H-th row), and none of the cells directly below the cells occupied by bar i is occupied by any bar, then bar i moves down by one cell. That is, if at that time bar i occupies the cells $(r, C_i), (r, C_i + 1), \dots, (r, C_i + L_i - 1)$ $(r < H)$, and the cell $(r + 1, C_i + j)$ is not occupied by any bar for all j $(0 \leq j \leq L_i - 1)$, then bar i now occupies $(r + 1, C_i), (r + 1, C_i + 1), \dots, (r + 1, C_i + L_i - 1)$.
- Otherwise, nothing happens.

Let $(R'_i, C_i), (R'_i, C_i + 1), \dots, (R'_i, C_i + L_i - 1)$ be the cells occupied by bar i at time $t = 10^{100}$. Find $R'_1, R'_2, \dots, R'_N$.

Constraints

- $1 \leq H, W \leq 2 \times 10^5$
- $1 \leq N \leq 2 \times 10^5$
- $1 \leq R_i \leq H$
- $1 \leq C_i \leq W$
- $1 \leq L_i \leq W - C_i + 1$
- In the initial state, there is no cell occupied by two different bars.
- All input values are integers.

Input

The input is given from Standard Input in the following format:

$$\begin{array}{lll} H & W & N \\ R_1 & C_1 & L_1 \\ R_2 & C_2 & L_2 \\ \vdots & & \\ R_N & C_N & L_N \end{array}$$

Output

Print N lines. The i-th line ($1 \leq i \leq N$) should contain R'_i.

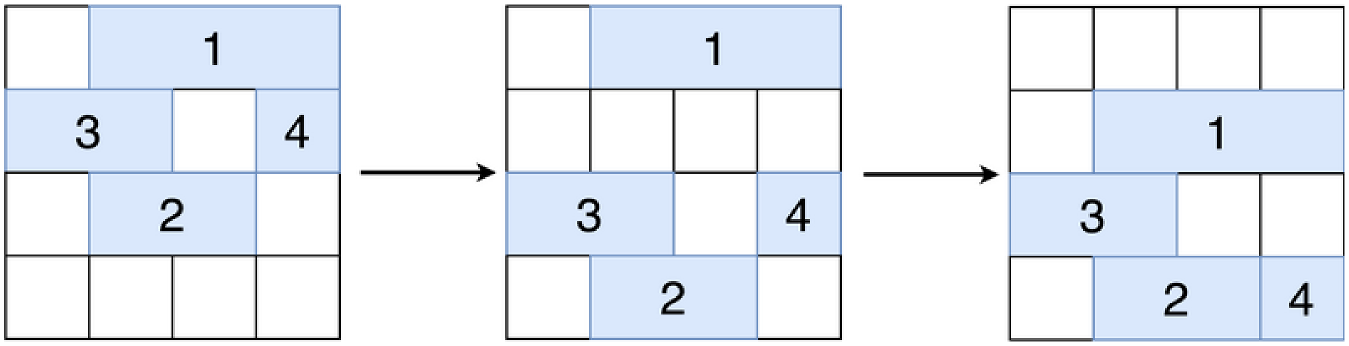
Sample Input 1

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

Sample Output 1

```
2
4
3
4
```

The following three diagrams represent the grid at times $t = 0, 1$, and 2 from left to right. Colored rectangles represent the bars, and the number inside each rectangle indicates its bar number.



The changes in the grid state are explained as follows:

- At $t = 0.5$:
  - $i = 1$: The cells directly below bar 1 are $(2, 2), (2, 3), (2, 4)$. Among these, $(2, 2)$ is occupied by bar 3 and $(2, 4)$ is occupied by bar 4, so nothing happens.
  - $i = 2$: The cells directly below bar 2 are $(4, 2), (4, 3)$, which are not occupied by any other bar, so bar 2 moves down by one cell.
  - $i = 3$: The cells directly below bar 3 are $(3, 1), (3, 2)$, which are not occupied by any other bar, so bar 3 moves down by one cell.
  - $i = 4$: The cell directly below bar 4 is $(3, 4)$, which is not occupied by any other bar, so bar 4 moves down by one cell.
- At $t = 1.5$:
  - $i = 1$: The cells directly below bar 1 are $(2, 2), (2, 3), (2, 4)$, which are not occupied by any other bar, so bar 1 moves down by one cell.
  - $i = 2$: Bar 2 is on the bottom row, so nothing happens.
  - $i = 3$: The cells directly below bar 3 are $(4, 1), (4, 2)$. Among these, $(4, 2)$ is occupied by bar 2, so nothing happens.
  - $i = 4$: The cell directly below bar 4 is $(4, 4)$, which is not occupied by any other bar, so bar 4 moves down by one cell.

At times $t = 2.5, 3.5, \ldots$, there is no bar such that the cells directly below it are all unoccupied, so nothing happens. Thus, the grid at time $t = 10^{100}$ is the same as at $t = 2$ (the rightmost diagram above).

Therefore, $R'_1 = 2, R'_2 = 4, R'_3 = 3, R'_4 = 4$.

Sample Input 2

```
382 382 3
3 3 3
8 8 8
2 2 2
```

Sample Output 2

```
382
382
381
```

Sample Input 3

```
5 10 8
2 2 1
4 3 1
4 8 2
1 2 2
2 5 3
5 4 3
4 5 2
1 5 2
```

Sample Output 3

```
5
5
5
4
3
5
4
2
```

G - Tile Distance 3

Time Limit: 2 sec / Memory Limit: 1024 MiB

Score: 575 points

Problem Statement

Tiles are laid out covering the two-dimensional coordinate plane.

Each tile is a rectangle, and for each integer triple (i, j, k) satisfying $0 \le k < K$, a corresponding tile is placed according to the following rules:

- When i and j have the same parity (both even or both odd), the tile corresponding to (i, j, k) covers the area where $iK \le x \le (i+1)K$ and $jK + k \le y \le jK + k + 1$.
- When i and j have different parity, the tile corresponding to (i, j, k) covers the area where $iK + k \le x \le iK + k + 1$ and $jK \le y \le (j+1)K$.

Two tiles are **adjacent** when their edges have a common segment of positive length.

Starting from the tile containing the point $(S_x + 0.5, S_y + 0.5)$, find the minimum number of times you need to move to an adjacent tile to reach the tile containing the point $(T_x + 0.5, T_y + 0.5)$.

There are T test cases; solve each of them.

Constraints

- $1 \le T \le 10^4$
- $2 \le K \le 10^{16}$
- $-10^{16} \le S_x, S_y, T_x, T_y \le 10^{16}$
- All input values are integers.

Input

The input is given from Standard Input in the following format:

$$\begin{array}{l} T \\ \mathrm{case}_1 \\ \vdots \\ \mathrm{case}_T \end{array}$$

Each case is given in the following format:

$$K \quad S_x \quad S_y \quad T_x \quad T_y$$

Output

Print T lines. The i-th line should contain the answer for the i-th test case.

Sample Input 1

```
3
3 -2 1 4 -1
4 8 8 0 2
5 -1000000000000 -1000000000000 1000000000000 1000000000000
```

Sample Output 1

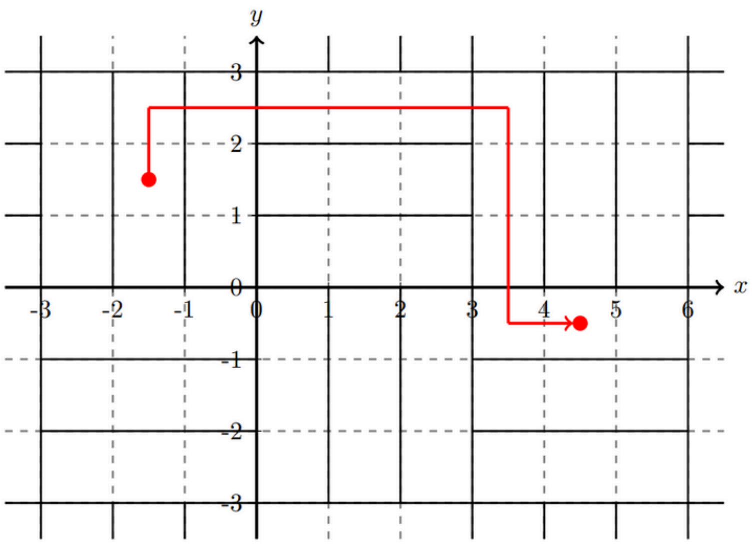
```
4
4
800000000000
```

Let us explain the first test case.

Let (i, j, k) denote the tile corresponding to integer triple (i, j, k).

$(-1.5, 1.5)$ is contained in tile $(-1, 0, 1)$, and $(4.5, -0.5)$ is contained in tile $(1, -1, 2)$.

For example, by moving from tile $(-1, 0, 1)$ to $(-1, 0, 2)$ to $(0, 0, 2)$ to $(1, 0, 0)$ to $(1, -1, 2)$, you can reach tile $(1, -1, 2)$ in four moves to an adjacent tile.


y
x
3
2
1
0
-1
-2
-3
-3
-2
-1
0
1
2
3
4
5
6