GPC-UPC DP I Contest

A. Complete the Projects (easy version)

2 seconds, 256 megabytes

The only difference between easy and hard versions is that you should complete all the projects in easy version but this is not necessary in hard version.

Polycarp is a very famous freelancer. His current rating is r units.

Some very rich customers asked him to complete some projects for their companies. To complete the i-th project, Polycarp needs to have at least a_i units of rating; after he completes this project, his rating will change by b_i (his rating will increase or decrease by b_i) (b_i can be positive or negative). Polycarp's rating should not fall below zero because then people won't trust such a low rated freelancer.

Is it possible to complete all the projects? Formally, write a program to check if such an order of the projects exists, that Polycarp has enough rating before starting each project, and he has non-negative rating after completing each project.

In other words, you have to check that there exists such an order of projects in which Polycarp will complete them, so he has enough rating before starting each project, and has non-negative rating after completing each project.

Input

The first line of the input contains two integers n and r ($1 \le n \le 100, 1 \le r \le 30000$) — the number of projects and the initial rating of Polycarp, respectively.

The next n lines contain projects, one per line. The i-th project is represented as a pair of integers a_i and b_i ($1 \le a_i \le 30000, -300 \le b_i \le 300$) — the rating required to complete the i-th project and the rating change after the project completion.

Output

Print "YES" or "NO".

input

```
3 4
4 6
10 -2
8 -1
```

output

```
YES
```

input

```
3 5
4 -5
4 -2
1 3
```

output

```
YES
```

input

```
4 4
5 2
5 -3
2 1
4 -2
```

output

```
YES
```

input

```
3 10
10 0
10 -10
30 0
```

output

```
NO
```

In the first example, the possible order is: 1, 2, 3.

In the second example, the possible order is: 2, 3, 1.

In the third example, the possible order is: 3, 1, 4, 2.

B. Complete the Projects (hard version)

2 seconds, 256 megabytes

The only difference between easy and hard versions is that you should complete all the projects in easy version but this is not necessary in hard version.

Polycarp is a very famous freelancer. His current rating is r units.

Some very rich customers asked him to complete some projects for their companies. To complete the i-th project, Polycarp needs to have at least a_i units of rating; after he completes this project, his rating will change by b_i (his rating will increase or decrease by b_i) (b_i can be positive or negative). Polycarp's rating should not fall below zero because then people won't trust such a low rated freelancer.

Polycarp can choose the order in which he completes projects. Furthermore, he can even skip some projects altogether.

To gain more experience (and money, of course) Polycarp wants to choose the subset of projects **having maximum possible size** and the order in which he will complete them, so he has enough rating before starting each project, and has non-negative rating after completing each project.

Your task is to calculate the maximum possible size of such subset of projects.

Input

The first line of the input contains two integers n and r ($1 \le n \le 100, 1 \le r \le 30000$) — the number of projects and the initial rating of Polycarp, respectively.

The next n lines contain projects, one per line. The i-th project is represented as a pair of integers a_i and b_i ($1 \le a_i \le 30000, -300 \le b_i \le 300$) — the rating required to complete the i-th project and the rating change after the project completion.

Output

Print one integer — the size of **the maximum possible** subset (possibly, empty) of projects Polycarp can choose.

input

```
3 4
4 6
10 -2
8 -1
```

output

```
3
```

input

```
5 20
45 -6
34 -15
10 34
1 27
40 -45
```

output

```
5
```

input

```
3 2
300 -300
1 299
1 123
```

output

```
3
```

C. Destroy it!

2 seconds, 256 megabytes

You are playing a computer card game called Splay the Sire. Currently you are struggling to defeat the final boss of the game.

The boss battle consists of n turns. During each turn, you will get several cards. Each card has two parameters: its cost c_i and damage d_i. You may play some of your cards during each turn in some sequence (you choose the cards and the exact order they are played), as long as **the total cost of the cards you play during the turn does not exceed 3**. After playing some (possibly zero) cards, you end your turn, and **all cards you didn't play are discarded**. Note that you can use each card **at most once**.

Your character has also found an artifact that boosts the damage of some of your actions: every 10-th card you play deals double damage.

What is the maximum possible damage you can deal during n turns?

Input

The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of turns.

Then n blocks of input follow, the i-th block representing the cards you get during the i-th turn.

Each block begins with a line containing one integer k_i ($1 \le k_i \le 2 \cdot 10^5$) — the number of cards you get during i-th turn. Then k_i lines follow, each containing two integers c_j and d_j ($1 \le c_j \le 3$, $1 \le d_j \le 10^9$) — the parameters of the corresponding card.

It is guaranteed that $\sum_{i=1}^{n} k_i \le 2 \cdot 10^5$.

Output

Print one integer — the maximum damage you may deal.

input

```
5
3
1 6
1 7
1 5
2
1 4
1 3
3
1 10
3 5
2 3
3
1 15
2 4
1 10
1
1 100
```

output

```
263
```

In the example test the best course of action is as follows:

During the first turn, play all three cards in any order and deal 18 damage.

During the second turn, play both cards and deal 7 damage.

During the third turn, play the first and the third card and deal 13 damage.

During the fourth turn, play the first and the third card and deal 25 damage.

During the fifth turn, play the only card, which will deal double damage (200).

D. Treasure Hunting

3 seconds, 256 megabytes

You are on the island which can be represented as a $n \times m$ table. The rows are numbered from 1 to n and the columns are numbered from 1 to m. There are k treasures on the island, the i-th of them is located at the position (r_i, c_i).

Initially you stand at the lower left corner of the island, at the position $(1, 1)$. If at any moment you are at the cell with a treasure, you can pick it up without any extra time. In one move you can move up (from (r, c) to $(r + 1, c)$), left (from (r, c) to $(r, c - 1)$), or right (from position (r, c) to $(r, c + 1)$). Because of the traps, you can't move down.

However, moving up is also risky. You can move up only if you are in a safe column. There are q safe columns: $b_1, b_2, \ldots, b_q$. You want to collect all the treasures as fast as possible. Count the minimum number of moves required to collect all the treasures.

Input

The first line contains integers n, m, k and q ($2 \le n, m, k, q \le 2 \cdot 10^5$, $q \le m$) — the number of rows, the number of columns, the number of treasures in the island and the number of safe columns.

Each of the next k lines contains two integers r_i, c_i, ($1 \le r_i \le n$, $1 \le c_i \le m$) — the coordinates of the cell with a treasure. All treasures are located in distinct cells.

The last line contains q distinct integers $b_1, b_2, \ldots, b_q$ ($1 \le b_i \le m$) — the indices of safe columns.

Output

Print the minimum number of moves required to collect all the treasures.

input

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

output

```
6
```

input

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

output

```
8
```

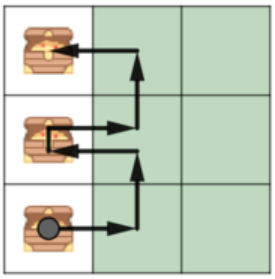
input

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

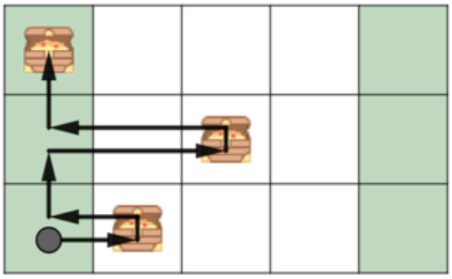
output

```
15
```

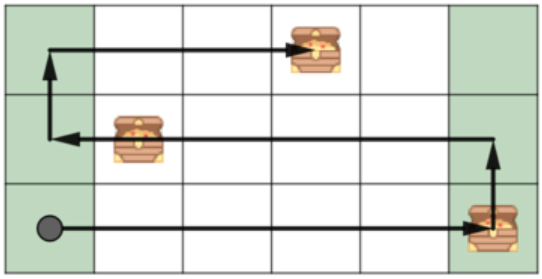
In the first example you should use the second column to go up, collecting in each row treasures from the first column.



In the second example, it is optimal to use the first column to go up.



In the third example, it is optimal to collect the treasure at cell $(1; 6)$, go up to row 2 at column 6, then collect the treasure at cell $(2; 2)$, go up to the top row at column 1 and collect the last treasure at cell $(3; 4)$. That's a total of 15 moves.



E. Subsequences (easy version)

2 seconds, 256 megabytes

The only difference between the easy and the hard versions is constraints.

A subsequence is a string that can be derived from another string by deleting some or no symbols without changing the order of the remaining symbols. Characters to be deleted are not required to go successively, there can be any gaps between them. For example, for the string "abaca" the following strings are subsequences: "abaca", "aba", "aaa", "a" and "" (empty string). But the following strings are not subsequences: "aabaca", "cb" and "bcaa".

You are given a string s consisting of n lowercase Latin letters.

In one move you can take **any** subsequence t of the given string and add it to the set S. The set S can't contain duplicates. This move costs $n - |t|$, where $|t|$ is the length of the added subsequence (i.e. the price equals to the number of the deleted characters).

Your task is to find out the minimum possible total cost to obtain a set S of size k or report that it is impossible to do so.

Input

The first line of the input contains two integers n and k ($1 \le n, k \le 100$) — the length of the string and the size of the set, correspondingly.

The second line of the input contains a string s consisting of n lowercase Latin letters.

Output

Print one integer — if it is impossible to obtain the set S of size k, print `-1`. Otherwise, print the minimum possible total cost to do it.

input
4 5 asdf

output
4

input
5 6 aaaaa

output
15

input
5 7 aaaaa

output
-1

input
10 100 ajihiushda

output
233

In the first example we can generate $S = \{$ "asdf", "asd", "adf", "asf", "sdf" $\}$. The cost of the first element in S is 0 and the cost of the others is 1. So the total cost of S is 4.

F. Subsequences (hard version)

2 seconds, 256 megabytes

The only difference between the easy and the hard versions is constraints.

A subsequence is a string that can be derived from another string by deleting some or no symbols without changing the order of the remaining symbols. Characters to be deleted are not required to go successively, there can be any gaps between them. For example, for the string "abaca" the following strings are subsequences: "abaca", "aba", "aaa", "a" and "" (empty string). But the following strings are not subsequences: "aabaca", "cb" and "bcaa".

You are given a string s consisting of n lowercase Latin letters.

In one move you can take **any** subsequence t of the given string and add it to the set S. The set S can't contain duplicates. This move costs $n - |t|$, where $|t|$ is the length of the added subsequence (i.e. the price equals to the number of the deleted characters).

Your task is to find out the minimum possible total cost to obtain a set S of size k or report that it is impossible to do so.

Input

The first line of the input contains two integers n and k ($1 \le n \le 100, 1 \le k \le 10^{12}$) — the length of the string and the size of the set, correspondingly.

The second line of the input contains a string s consisting of n lowercase Latin letters.

Output

Print one integer — if it is impossible to obtain the set S of size k, print `-1`. Otherwise, print the minimum possible total cost to do it.

input
4 5 asdf

output
4

input
5 6 aaaaa

output
15

input
5 7 aaaaa
output
-1

input
10 100 ajihiushda
output
233

In the first example we can generate S = { "asdf", "asd", "adf", "asf", "sdf" }. The cost of the first element in S is 0 and the cost of the others is 1. So the total cost of S is 4.

G. Short Colorful Strip

3 seconds, 256 megabytes

This is the first subtask of problem F. The only differences between this and the second subtask are the constraints on the value of m and the time limit. You need to solve both subtasks in order to hack this one.

There are $n + 1$ distinct colours in the universe, numbered 0 through n. There is a strip of paper m centimetres long initially painted with colour 0.

Alice took a brush and painted the strip using the following process. For each i from 1 to n, **in this order**, she picks two integers $0 \le a_i < b_i \le m$, such that the segment $[a_i, b_i]$ is currently painted with a **single** colour, and repaints it with colour i.

Alice chose the segments in such a way that each centimetre is now painted in some colour other than 0. Formally, the segment $[i - 1, i]$ is painted with colour c_i ($c_i \neq 0$). Every colour other than 0 is visible on the strip.

Count the number of different pairs of sequences $\{a_i\}_{i=1}^{n}$, $\{b_i\}_{i=1}^{n}$ that result in this configuration.

Since this number may be large, output it modulo 998244353.

Input

The first line contains a two integers n, m ($1 \le n \le 500$, $n = m$) — the number of colours excluding the colour 0 and the length of the paper, respectively.

The second line contains m space separated integers $c_1, c_2, \ldots, c_m$ ($1 \le c_i \le n$) — the colour visible on the segment $[i - 1, i]$ after the process ends. It is guaranteed that for all j between 1 and n there is an index k such that $c_k = j$.

Note that since in this subtask $n = m$, this means that c is a permutation of integers 1 through n.

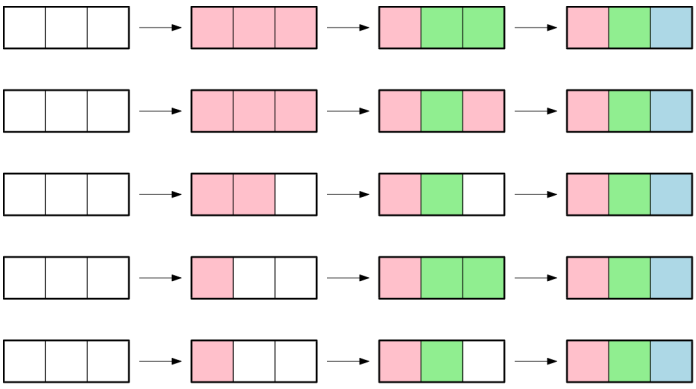
Output

Output a single integer — the number of ways Alice can perform the painting, modulo 998244353.

input
3 3 1 2 3
output
5

input
7 7 4 5 1 6 2 3 7
output
165

In the first example, there are 5 ways, all depicted in the figure below. Here, 0 is white, 1 is red, 2 is green and 3 is blue.



Below is an example of a painting process that is not valid, as in the second step the segment 1 3 is not single colour, and thus may not be repainted with colour 2.

H. Long Colorful Strip

6 seconds, 256 megabytes

This is the second subtask of problem F. The only differences between this and the first subtask are the constraints on the value of m and the time limit. It is sufficient to solve this subtask in order to hack it, but you need to solve both subtasks in order to hack the first one.

There are $n + 1$ distinct colours in the universe, numbered 0 through n. There is a strip of paper m centimetres long initially painted with colour 0.

Alice took a brush and painted the strip using the following process. For each i from 1 to n, **in this order**, she picks two integers $0 \le a_i < b_i \le m$, such that the segment $[a_i, b_i]$ is currently painted with a **single** colour, and repaints it with colour i.

Alice chose the segments in such a way that each centimetre is now painted in some colour other than 0. Formally, the segment $[i - 1, i]$ is painted with colour c_i ($c_i \neq 0$). Every colour other than 0 is visible on the strip.

Count the number of different pairs of sequences $\{a_i\}_{i=1}^{n}$, $\{b_i\}_{i=1}^{n}$ that result in this configuration.

Since this number may be large, output it modulo 998244353.

Input

The first line contains a two integers n, m ($1 \le n \le 500$, $n \le m \le 10^6$) — the number of colours excluding the colour 0 and the length of the paper, respectively.

The second line contains m space separated integers $c_1, c_2, \ldots, c_m$ ($1 \le c_i \le n$) — the colour visible on the segment $[i - 1, i]$ after the process ends. It is guaranteed that for all j between 1 and n there is an index k such that $c_k = j$.

Output

Output a single integer — the number of ways Alice can perform the painting, modulo 998244353.

input
3 3 1 2 3
output
5

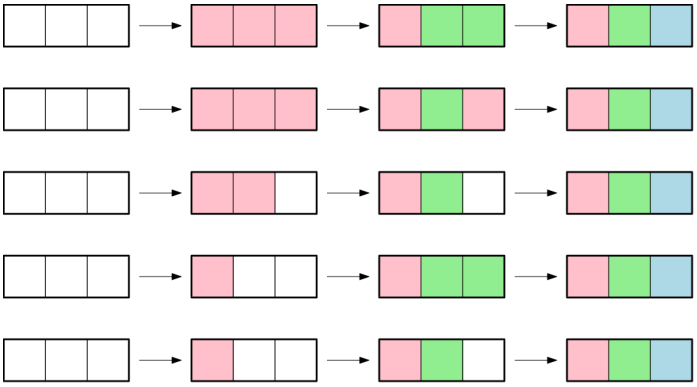
input
2 3 1 2 1
output
1

input
2 3 2 1 2
output
0

input
7 7 4 5 1 6 2 3 7
output
165

input
8 17 1 3 2 2 7 8 2 5 5 4 4 4 1 1 6 1 1
output
20

In the first example, there are 5 ways, all depicted in the figure below. Here, 0 is white, 1 is red, 2 is green and 3 is blue.



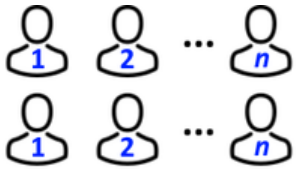
Below is an example of a painting process that is not valid, as in the second step the segment 1 3 is not single colour, and thus may not be repainted with colour 2.

In the second example, Alice must first paint segment 0 3 with colour 1 and then segment 1 2 with colour 2.

I. Basketball Exercise

2 seconds, 256 megabytes

Finally, a basketball court has been opened in SIS, so Demid has decided to hold a basketball exercise session. $2 \cdot n$ students have come to Demid's exercise session, and he lined up them into two rows of the same size (there are exactly n people in each row). Students are numbered from 1 to n in each row in order from left to right.



Now Demid wants to choose a team to play basketball. He will choose players from left to right, and the index of each chosen player (excluding the first one **taken**) will be strictly greater than the index of the previously chosen player. To avoid giving preference to one of the rows, Demid chooses students in such a way that no consecutive chosen students belong to the same row. The first student can be chosen among all $2n$ students (there are no additional constraints), and a team can consist of any number of students.

Demid thinks, that in order to compose a perfect team, he should choose students in such a way, that the total height of all chosen students is maximum possible. Help Demid to find the maximum possible total height of players in a team he can choose.

Input

The first line of the input contains a single integer n ($1 \le n \le 10^5$) — the number of students in each row.

The second line of the input contains n integers $h_{1,1}, h_{1,2}, \ldots, h_{1,n}$ ($1 \le h_{1,i} \le 10^9$), where $h_{1,i}$ is the height of the i-th student in the first row.

The third line of the input contains n integers $h_{2,1}, h_{2,2}, \ldots, h_{2,n}$ ($1 \le h_{2,i} \le 10^9$), where $h_{2,i}$ is the height of the i-th student in the second row.

Output

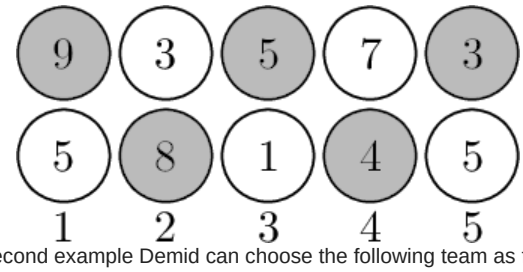
Print a single integer — the maximum possible total height of players in a team Demid can choose.

input
5 9 3 5 7 3 5 8 1 4 5
output
29

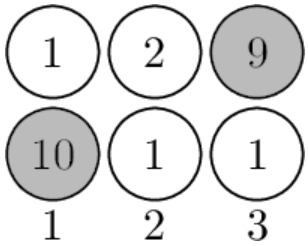
input
3 1 2 9 10 1 1
output
19

input
1 7 4
output
7

In the first example Demid can choose the following team as follows:



In the second example Demid can choose the following team as follows:



J. Array Beauty

5 seconds, 256 megabytes

Let's call beauty of an array $b_1, b_2, \ldots, b_n$ ($n > 1$) — $\min\limits_{1 \le i < j \le n} |b_i - b_j|$.

You're given an array $a_1, a_2, \ldots a_n$ and a number k. Calculate the sum of beauty over all subsequences of the array of length exactly k. As this number can be very large, output it modulo 998244353.

A sequence a is a subsequence of an array b if a can be obtained from b by deletion of several (possibly, zero or all) elements.

Input
The first line contains integers n, k ($2 \le k \le n \le 1000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^5$).

Output
Output one integer — the sum of beauty over all subsequences of the array of length exactly k. As this number can be very large, output it modulo 998244353.

input
4 3 1 7 3 5
output
8

input
5 5 1 10 100 1000 10000
output
9

In the first example, there are 4 subsequences of length 3 — $[1, 7, 3]$, $[1, 3, 5]$, $[7, 3, 5]$, $[1, 7, 5]$, each of which has beauty 2, so answer is 8.

In the second example, there is only one subsequence of length 5 — the whole array, which has the beauty equal to $|10 - 1| = 9$.

K. Carnival Slots

1.5 seconds, 256 megabytes

You go to the carnival and come across a nice little game. The carnival worker shows you the setup of the game, which can be represented as a 2-dimensional grid g with r rows and c columns. You are given the opportunity to change around the grid and maximize your score before the worker drops several balls into each column.

A cell in the i^{th} row (from top) and the j^{th} column (from left) is denoted by (i, j), and can be one of three different types of cells:

1. '.'; a ball that enters this cell will go to cell $(i + 1, j)$.
2. '\'; a ball that enters this cell will go to cell $(i + 1, j + 1)$.
3. '/'; a ball that enters this cell will go to cell $(i + 1, j - 1)$.

You may change a cell of type 2 and 3 to any of the three types, and you can change as many cells as you want. Cells of type 1 can't be changed.

Under the grid is aligned c buckets, where the i^{th} bucket is below the i^{th} column.

Each of the c buckets contains a score. For every ball that falls into a bucket, the score on that bucket is added to your total score and that ball stops. A ball that doesn't fall into a bucket gets a score of 0.

You are given how many balls the worker will drop into each column. Balls are dropped one after the another such that no two balls will collide. After making the changes, what is the maximum score you can achieve?

Your only condition for the grid g is that it's not allowed to have two adjacent cells in one row such that the left one is '\' and the right one is '/'.

Input
The first line of input contains a single integer T ($1 \le T \le 5300$), the number of test cases.

The first line of each test case contains two space-separated integers r and c ($1 \le r, c \le 500$), the dimensions of the grid.

The following line contains c space-separated integers $b_1, b_2, \ldots, b_c$ ($0 \le b_i \le 10^8$), where b_i is the number of balls dropped into the i^{th} column.

Each of the following r lines contains c characters, representing the grid. Each character is either '.', '\', or '/'.

The last line of each test case contains c space-separated integers $s_1, s_2, \ldots, s_c$ ($0 \le s_i \le 10^8$), where s_i is the score added after one ball drops into the i^{th} bucket.

The sum of $r \times c$ over all test cases doesn't exceed 4×10^6.

Output
For each test case, print on a single line the maximum score possible.

input
2 3 3 1 2 1 ../ ./. \.. 10 5 20 1 2 100000000 100000000 .. 1 100000000
output
70 10000000100000000

L. Fascination Street

2 seconds, 1024 megabytes

A street named *Fascination Street* is divided into N equal length of blocks. For each block i ($1 \le i \le N$), it has block $i - 1$ in its left side if $i > 1$, and block $i + 1$ in its right side if $i < N$.

Unlike its name, the street is infamous to be a dark and eerie place in the night. To solve this, Robert decided to install the streetlight for some of the blocks. The cost of installing a streetlight for i-th block is W_i, and the total cost is the sum of each installation cost. After installing, every block should either have a streetlight, or have a streetlight in it's left or right block.

Robert also has some tricks to reduce the cost. Before installing the streetlight, Robert selects two distinct blocks i and j, and exchange their position. After the operation, the cost of installation is exchanged. In other words, the operation simply swaps the value of W_i and W_j. This operation have no cost, but Robert can only perform it at most K times.

Now, given the array W and the maximum possible number of operations K, you should find the minimum cost of lighting the whole street.

Input
The first line contains two space-separated integers N, K. N is the number of blocks, and K is the maximum possible number of operations. ($1 \le N \le 250000, 0 \le K \le 9$)

The second line contains N space-separated integers $W_1, W_2 \cdots W_N$, where W_i is the cost of installing a streetlight for i-th block. ($0 \le W_i \le 10^9$)

Output
Print a single integer which contains the minimum cost of lighting the whole street.

input
5 0 1 3 10 3 1

output

```
4
```

input

```
5 1
1 3 10 3 1
```

output

```
2
```

input

```
12 0
317 448 258 208 284 248 315 367 562 500 426 390
```

output

```
1525
```

input

```
12 2
317 448 258 208 284 248 315 367 562 500 426 390
```

output

```
1107
```

M. Zeros and Ones

15 seconds, 1024 megabytes

Given four integers x, y, M and K. You need to count the number of integers d that satisfy the following rules:

1. The binary representation of d should consist of x ones and y zeroes without leading zeros.
2. d modulo M is greater than or equal to K.

Input

The first line of the input is the number of test cases T. Each test case consists of four integers x, y, M and K, where $1 \le M \le 10^9$, $1 \le x + y \le 42$, $x \ge 1$, $y \ge 0$ and $0 \le K < M$.

Output

For each test case output a single line with the number of integers d that satisfy the rules.

input

```
1
2 2 8 2
```

output

```
2
```

N. Make The Fence Great Again

2 seconds, 256 megabytes

You have a fence consisting of n vertical boards. The width of each board is 1. The height of the i-th board is a_i. You think that the fence is *great* if there is no pair of adjacent boards having the same height. More formally, the fence is great if and only if for all indices from 2 to n, the condition $a_{i-1} \neq a_i$ holds.

Unfortunately, it is possible that now your fence is not great. But you can change it! You can increase the length of the i-th board by 1, but you have to pay b_i rubles for it. The length of each board can be increased any number of times (possibly, zero).

Calculate the minimum number of rubles you have to spend to make the fence great again!

You have to answer q independent queries.

Input

The first line contains one integer q ($1 \le q \le 3 \cdot 10^5$) — the number of queries.

The first line of each query contains one integers n ($1 \le n \le 3 \cdot 10^5$) — the number of boards in the fence.

The following n lines of each query contain the descriptions of the boards. The i-th line contains two integers a_i and b_i ($1 \le a_i, b_i \le 10^9$) — the length of the i-th board and the price for increasing it by 1, respectively.

It is guaranteed that sum of all n over all queries not exceed $3 \cdot 10^5$.

It is guaranteed that answer to each query will not exceed 10^{18}.

Output

For each query print one integer — the minimum number of rubles you have to spend to make the fence great.

input

```
3
3
2 4
2 1
3 5
3
2 3
2 10
2 6
4
1 7
3 3
2 6
1000000000 2
```

output

```
2
9
0
```

In the first query you have to increase the length of second board by 2. So your total costs if $2 \cdot b_2 = 2$.

In the second query you have to increase the length of first board by 1 and the length of third board by 1. So your total costs if $1 \cdot b_1 + 1 \cdot b_3 = 9$.

In the third query the fence is great initially, so you don't need to spend rubles.

O. Make Product Equal One

1 second, 256 megabytes

You are given n numbers $a_1, a_2, \ldots, a_n$. With a cost of one coin you can perform the following operation:

Choose one of these numbers and add or subtract 1 from it.

In particular, we can apply this operation to the same number several times.

We want to make the product of all these numbers equal to 1, in other words, we want $a_1 \cdot a_2 \ldots \cdot a_n = 1$.

For example, for $n = 3$ and numbers $[1, -3, 0]$ we can make product equal to 1 in 3 coins: add 1 to second element, add 1 to second element again, subtract 1 from third element, so that array becomes $[1, -1, -1]$. And $1 \cdot (-1) \cdot (-1) = 1$.

What is the minimum cost we will have to pay to do that?

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the number of numbers.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$) — the numbers.

Output

Output a single number — the minimal number of coins you need to pay to make the product equal to 1.

input

```
2
-1 1
```

output
2

input
4 0 0 0 0
output
4

input
5 -5 -3 5 3 0
output
13

In the first example, you can change 1 to -1 or -1 to 1 in 2 coins.

In the second example, you have to apply at least 4 operations for the product not to be 0.

In the third example, you can change -5 to -1 in 4 coins, -3 to -1 in 2 coins, 5 to 1 in 4 coins, 3 to 1 in 2 coins, 0 to 1 in 1 coin.

P. Polycarp and Div 3

3 seconds, 256 megabytes

Polycarp likes numbers that are divisible by 3.

He has a huge number s. Polycarp wants to cut from it the maximum number of numbers that are divisible by 3. To do this, he makes an arbitrary number of vertical cuts between pairs of adjacent digits. As a result, after m such cuts, there will be $m + 1$ parts in total. Polycarp analyzes each of the obtained numbers and finds the number of those that are divisible by 3.

For example, if the original number is $s = 3121$, then Polycarp can cut it into three parts with two cuts: $3|1|21$. As a result, he will get two numbers that are divisible by 3.

Polycarp can make an arbitrary number of vertical cuts, where each cut is made between a pair of adjacent digits. The resulting numbers cannot contain extra leading zeroes (that is, the number can begin with `0` if and only if this number is exactly one character '0'). For example, `007`, `01` and `00099` are not valid numbers, but `90`, `0` and `10001` are valid.

What is the maximum number of numbers divisible by 3 that Polycarp can obtain?

Input

The first line of the input contains a positive integer s. The number of digits of the number s is between 1 and $2 \cdot 10^5$, inclusive. The first (leftmost) digit is not equal to `0`.

Output

Print the maximum number of numbers divisible by 3 that Polycarp can get by making vertical cuts in the given number s.

input
3121
output
2

input
6
output
1

input
1000000000000000000000000000000000
output
33

input
201920181
output
4

In the first example, an example set of optimal cuts on the number is `3|1|21`.

In the second example, you do not need to make any cuts. The specified number `6` forms one number that is divisible by 3.

In the third example, cuts must be made between each pair of digits. As a result, Polycarp gets one digit `1` and 33 digits `0`. Each of the 33 digits `0` forms a number that is divisible by 3.

In the fourth example, an example set of optimal cuts is `2|0|1|9|201|81`. The numbers 0, 9, 201 and 81 are divisible by 3.

Q. Consecutive Subsequence

2 seconds, 256 megabytes

You are given an integer array of length n.

You have to choose some subsequence of this array of maximum length such that this subsequence forms a increasing sequence of consecutive integers. In other words the required sequence should be equal to $[x, x + 1, \ldots, x + k - 1]$ for some value x and length k.

Subsequence of an array can be obtained by erasing some (possibly zero) elements from the array. You can erase any elements, not necessarily going successively. The remaining elements preserve their order. For example, for the array $[5, 3, 1, 2, 4]$ the following arrays are subsequences: $[3]$, $[5, 3, 1, 2, 4]$, $[5, 1, 4]$, but the array $[1, 3]$ is not.

Input

The first line of the input containing integer number n ($1 \le n \le 2 \cdot 10^5$) — the length of the array. The second line of the input containing n integer numbers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the array itself.

Output

On the first line print k — the maximum length of the subsequence of the given array that forms an increasing sequence of consecutive integers.

On the second line print the sequence of the indices of the **any** maximum length subsequence of the given array that forms an increasing sequence of consecutive integers.

input
7 3 3 4 7 5 6 8
output
4 2 3 5 6

input
6 1 3 5 2 4 6
output
2 1 4

input
4 10 9 8 7
output
1 1

input
9 6 7 8 3 4 5 9 10 11
output
6 1 2 3 7 8 9

All valid answers for the first example (as sequences of indices):

- [1, 3, 5, 6]
- [2, 3, 5, 6]

All valid answers for the second example:

- [1, 4]
- [2, 5]
- [3, 6]

All valid answers for the third example:

- [1]
- [2]
- [3]
- [4]

All valid answers for the fourth example:

- [1, 2, 3, 7, 8, 9]

R. Cutting

2 seconds, 256 megabytes

There are a lot of things which could be cut — trees, paper, "the rope". In this problem you are going to cut a sequence of integers.

There is a sequence of integers, which contains the equal number of even and odd numbers. Given a limited budget, you need to make maximum possible number of cuts such that each resulting segment will have the same number of odd and even integers.

Cuts separate a sequence to continuous (contiguous) segments. You may think about each cut as a break between two adjacent elements in a sequence. So after cutting each element belongs to exactly one segment. Say, [4, 1, 2, 3, 4, 5, 4, 4, 5, 5] → two cuts → [4, 1 | 2, 3, 4, 5 | 4, 4, 5, 5]. On each segment the number of even elements should be equal to the number of odd elements.

The cost of the cut between x and y numbers is $|x - y|$ bitcoins. Find the maximum possible number of cuts that can be made while spending no more than B bitcoins.

Input

First line of the input contains an integer n ($2 \le n \le 100$) and an integer B ($1 \le B \le 100$) — the number of elements in the sequence and the number of bitcoins you have.

Second line contains n integers: $a_1, a_2, ..., a_n$ ($1 \le a_i \le 100$) — elements of the sequence, which contains the equal number of even and odd numbers

Output

Print the maximum possible number of cuts which can be made while spending no more than B bitcoins.

input
6 4 1 2 5 10 15 20
output
1

input
4 10 1 3 2 4
output
0

input
6 100 1 2 3 4 5 6
output
2

In the first sample the optimal answer is to split sequence between 2 and 5. Price of this cut is equal to 3 bitcoins.

In the second sample it is not possible to make even one cut even with unlimited number of bitcoins.

In the third sample the sequence should be cut between 2 and 3, and between 4 and 5. The total price of the cuts is 1 + 1 = 2 bitcoins.

S. Party Lemonade

1 second, 256 megabytes

A New Year party is not a New Year party without lemonade! As usual, you are expecting a lot of guests, and buying lemonade has already become a pleasant necessity.

Your favorite store sells lemonade in bottles of n different volumes at different costs. A single bottle of type i has volume 2^{i-1} liters and costs c_i roubles. The number of bottles of each type in the store can be considered infinite.

You want to buy at least L liters of lemonade. How many roubles do you have to spend?

Input

The first line contains two integers n and L ($1 \le n \le 30$; $1 \le L \le 10^9$) — the number of types of bottles in the store and the required amount of lemonade in liters, respectively.

The second line contains n integers $c_1, c_2, ..., c_n$ ($1 \le c_i \le 10^9$) — the costs of bottles of different types.

Output

Output a single integer — the smallest number of roubles you have to pay in order to buy at least L liters of lemonade.

input
4 12 20 30 70 90
output
150

input
4 3 10000 1000 100 10
output
10

input
4 3 10 100 1000 10000
output
30

input
5 787787787 123456789 234567890 345678901 456789012 987654321
output
44981600785557577

In the first example you should buy one 8-liter bottle for 90 roubles and two 2-liter bottles for 30 roubles each. In total you'll get 12 liters of lemonade for just 150 roubles.

In the second example, even though you need only 3 liters, it's cheaper to buy a single 8-liter bottle for 10 roubles.

In the third example it's best to buy three 1-liter bottles for 10 roubles each, getting three liters for 30 roubles.

T. Permute Digits

1 second, 256 megabytes

You are given two positive integer numbers a and b. Permute (change order) of the digits of a to construct maximal number not exceeding b. No number in input and/or output can start with the digit 0.

It is allowed to leave a as it is.

Input

The first line contains integer a ($1 \le a \le 10^{18}$). The second line contains integer b ($1 \le b \le 10^{18}$). Numbers don't have leading zeroes. It is guaranteed that answer exists.

Output

Print the maximum possible number that is a permutation of digits of a and is not greater than b. The answer can't have any leading zeroes. It is guaranteed that the answer exists.

The number in the output should have exactly the same length as number a. It should be a permutation of digits of a.

input
123 222
output
213

input
3921 10000
output
9321

input
4940 5000
output
4940

U. Pride

2 seconds, 256 megabytes

You have an array a with length n, you can perform operations. Each operation is like this: choose two **adjacent** elements from a, say x and y, and replace one of them with $gcd(x, y)$, where gcd denotes the greatest common divisor.

What is the minimum number of operations you need to make all of the elements equal to 1?

Input

The first line of the input contains one integer n ($1 \le n \le 2000$) — the number of elements in the array.

The second line contains n space separated integers $a_1, a_2, ..., a_n$ ($1 \le a_i \le 10^9$) — the elements of the array.

Output

Print -1, if it is impossible to turn all numbers to 1. Otherwise, print the minimum number of operations needed to make all numbers equal to 1.

input
5 2 2 3 4 6
output
5

input
4 2 4 6 8
output
-1

input
3 2 6 9
output
4

In the first sample you can turn all numbers to 1 using the following 5 moves:

- $[2, 2, 3, 4, 6]$.
- $[2, 1, 3, 4, 6]$
- $[2, 1, 3, 1, 6]$
- $[2, 1, 1, 1, 6]$
- $[1, 1, 1, 1, 6]$
- $[1, 1, 1, 1, 1]$

We can prove that in this case it is not possible to make all numbers one using less than 5 moves.

V. Almost Identity Permutations

2 seconds, 256 megabytes

A permutation p of size n is an array such that every integer from 1 to n occurs exactly once in this array.

Let's call a permutation an *almost identity permutation* iff there exist at least $n - k$ indices i ($1 \le i \le n$) such that $p_i = i$.

Your task is to count the number of *almost identity* permutations for given numbers n and k.

Input

The first line contains two integers n and k ($4 \le n \le 1000$, $1 \le k \le 4$).

Output

Print the number of *almost identity* permutations for given n and k.

input
4 1
output
1

input
4 2
output
7

input
5 3
output
31

input
5 4

output
76

W. Star sky

2 seconds, 256 megabytes

The Cartesian coordinate system is set in the sky. There you can see n stars, the i-th has coordinates (x_i, y_i), a maximum brightness c, equal for all stars, and an initial brightness s_i ($0 \le s_i \le c$).

Over time the stars twinkle. At moment 0 the i-th star has brightness s_i. Let at moment t some star has brightness x. Then at moment $(t+1)$ this star will have brightness $x+1$, if $x+1 \le c$, and 0, otherwise.

You want to look at the sky q times. In the i-th time you will look at the moment t_i and you will see a rectangle with sides parallel to the coordinate axes, the lower left corner has coordinates (x_{1i}, y_{1i}) and the upper right — (x_{2i}, y_{2i}). For each view, you want to know the total brightness of the stars lying in the viewed rectangle.

A star lies in a rectangle if it lies on its border or lies strictly inside it.

Input

The first line contains three integers n, q, c ($1 \le n, q \le 10^5$, $1 \le c \le 10$) — the number of the stars, the number of the views and the maximum brightness of the stars.

The next n lines contain the stars description. The i-th from these lines contains three integers x_i, y_i, s_i ($1 \le x_i, y_i \le 100$, $0 \le s_i \le c \le 10$) — the coordinates of i-th star and its initial brightness.

The next q lines contain the views description. The i-th from these lines contains five integers $t_i, x_{1i}, y_{1i}, x_{2i}, y_{2i}$ ($0 \le t_i \le 10^9$, $1 \le x_{1i} < x_{2i} \le 100$, $1 \le y_{1i} < y_{2i} \le 100$) — the moment of the i-th view and the coordinates of the viewed rectangle.

Output

For each view print the total brightness of the viewed stars.

input
2 3 3 1 1 1 3 2 0 2 1 1 2 2 0 2 1 4 5 5 1 1 5 5
output
3 0 3

input
3 4 5 1 1 2 2 3 0 3 3 1 0 1 1 100 100 1 2 2 4 4 2 2 1 4 7 1 50 50 51 51
output
3 3 5 0

Let's consider the first example.

At the first view, you can see only the first star. At moment 2 its brightness is 3, so the answer is 3.

At the second view, you can see only the second star. At moment 0 its brightness is 0, so the answer is 0.

At the third view, you can see both stars. At moment 5 brightness of the first is 2, and brightness of the second is 1, so the answer is 3.

X. Sagheer, the Hausmeister

1 second, 256 megabytes

Some people leave the lights at their workplaces on when they leave that is a waste of resources. As a hausmeister of DHBW, Sagheer waits till all students and professors leave the university building, then goes and turns all the lights off.

The building consists of n floors with stairs at the left and the right sides. Each floor has m rooms on the same line with a corridor that connects the left and right stairs passing by all the rooms. In other words, the building can be represented as a rectangle with n rows and $m+2$ columns, where the first and the last columns represent the stairs, and the m columns in the middle represent rooms.

Sagheer is standing at the ground floor at the left stairs. He wants to turn all the lights off in such a way that he will not go upstairs until all lights in the floor he is standing at are off. Of course, Sagheer must visit a room to turn the light there off. It takes one minute for Sagheer to go to the next floor using stairs or to move from the current room/stairs to a neighboring room/stairs on the same floor. It takes no time for him to switch the light off in the room he is currently standing in. Help Sagheer find the minimum total time to turn off all the lights.

Note that Sagheer does not have to go back to his starting position, and he does not have to visit rooms where the light is already switched off.

Input

The first line contains two integers n and m ($1 \le n \le 15$ and $1 \le m \le 100$) — the number of floors and the number of rooms in each floor, respectively.

The next n lines contains the building description. Each line contains a binary string of length $m+2$ representing a floor (the left stairs, then m rooms, then the right stairs) where 0 indicates that the light is off and 1 indicates that the light is on. The floors are listed from top to bottom, so that the last line represents the ground floor.

The first and last characters of each string represent the left and the right stairs, respectively, so they are always 0.

Output

Print a single integer — the minimum total time needed to turn off all the lights.

input
2 2 0010 0100
output
5

input
3 4 001000 000010 000010
output
12

input
4 3 01110 01110 01110 01110
output
18

In the first example, Sagheer will go to room 1 in the ground floor, then he will go to room 2 in the second floor using the left or right stairs.

In the second example, he will go to the fourth room in the ground floor, use right stairs, go to the fourth room in the second floor, use right stairs again, then go to the second room in the last floor.

In the third example, he will walk through the whole corridor alternating between the left and right stairs at each floor.

Y. Odd sum

1 second, 256 megabytes

You are given sequence $a_1, a_2, ..., a_n$ of integer numbers of length n. Your task is to find such subsequence that its sum is odd and maximum among all such subsequences. It's guaranteed that given sequence contains subsequence with odd sum.

Subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements.

You should write a program which finds sum of the best subsequence.

Input

The first line contains integer number n ($1 \le n \le 10^5$).

The second line contains n integer numbers $a_1, a_2, ..., a_n$ ($-10^4 \le a_i \le 10^4$). The sequence contains at least one subsequence with odd sum.

Output

Print sum of resulting subseqeuence.

input
4 -2 2 -3 1
output
3

input
3 2 -5 -3
output
-1

In the first example sum of the second and the fourth elements is 3.

Z. Cormen — The Best Friend Of a Man

1 second, 256 megabytes

Recently a dog was bought for Polycarp. The dog's name is Cormen. Now Polycarp has a lot of troubles. For example, Cormen likes going for a walk.

Empirically Polycarp learned that the dog needs at least k walks for any two consecutive days in order to feel good. For example, if $k = 5$ and yesterday Polycarp went for a walk with Cormen 2 times, today he has to go for a walk at least 3 times.

Polycarp analysed all his affairs over the next n days and made a sequence of n integers $a_1, a_2, ..., a_n$, where a_i is the number of times Polycarp will walk with the dog on the i-th day while doing all his affairs (for example, he has to go to a shop, throw out the trash, etc.).

Help Polycarp determine the minimum number of walks he needs to do additionaly in the next n days so that Cormen will feel good during all the n days. You can assume that on the day before the first day and on the day after the n-th day Polycarp will go for a walk with Cormen exactly k times.

Write a program that will find the minumum number of additional walks and the appropriate schedule — the sequence of integers $b_1, b_2, ..., b_n$ ($b_i \ge a_i$), where b_i means the total number of walks with the dog on the i-th day.

Input

The first line contains two integers n and k ($1 \le n, k \le 500$) — the number of days and the minimum number of walks with Cormen for any two consecutive days.

The second line contains integers $a_1, a_2, ..., a_n$ ($0 \le a_i \le 500$) — the number of walks with Cormen on the i-th day which Polycarp has already planned.

Output

In the first line print the smallest number of additional walks that Polycarp should do during the next n days so that Cormen will feel good during all days.

In the second line print n integers $b_1, b_2, ..., b_n$, where b_i — the total number of walks on the i-th day according to the found solutions ($a_i \le b_i$ for all i from 1 to n). If there are multiple solutions, print any of them.

input
3 5 2 0 1
output
4 2 3 2

input
3 1 0 0 0
output
1 0 1 0

input
4 6 2 4 3 5
output
0 2 4 3 5