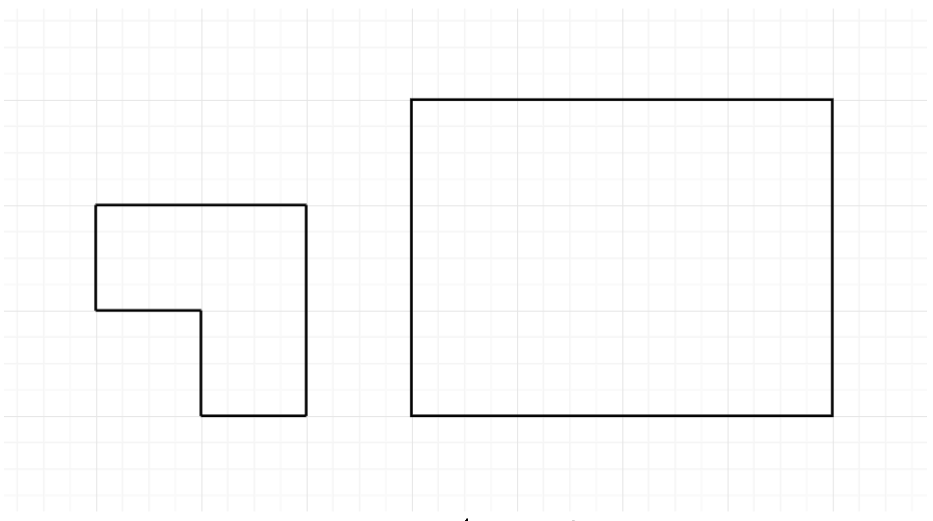
Topic Stream Mashup: Dynamic Programming

A. Filling Shapes

1 second, 256 megabytes

You have a given integer n. Find the number of ways to fill all $3 \times n$ tiles with the shape described in the picture below. Upon filling, no empty spaces are allowed. Shapes cannot overlap.



This picture describes when $n = 4$. The left one is the shape and the right one is $3 \times n$ tiles.

Input

The only line contains one integer n $(1 \leq n \leq 60)$ — the length.

Output

Print the number of ways to fill.

input
```
4
```
output
```
4
```

input
```
1
```
output
```
0
```

In the first example, there are 4 possible cases of filling.

In the second example, you cannot fill the shapes in 3×1 tiles.

B. Yet Another Broken Keyboard

2 seconds, 256 megabytes

Recently, Norge found a string $s = s_1 s_2 \ldots s_n$ consisting of n lowercase Latin letters. As an exercise to improve his typing speed, he decided to type all substrings of the string s. Yes, all $\frac{n(n+1)}{2}$ of them!

A substring of s is a non-empty string $x = s[a \ldots b] = s_a s_{a+1} \ldots s_b$ ($1 \leq a \leq b \leq n$). For example, "auto" and "ton" are substrings of "automaton".

Shortly after the start of the exercise, Norge realized that his keyboard was broken, namely, he could use only k Latin letters $c_1, c_2, \ldots, c_k$ out of 26.

After that, Norge became interested in how many substrings of the string s he could still type using his broken keyboard. Help him to find this number.

Input

The first line contains two space-separated integers n and k ($1 \leq n \leq 2 \cdot 10^5, 1 \leq k \leq 26$) — the length of the string s and the number of Latin letters still available on the keyboard.

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

The third line contains k space-separated distinct lowercase Latin letters $c_1, c_2, \ldots, c_k$ — the letters still available on the keyboard.

Output

Print a single number — the number of substrings of s that can be typed using only available letters $c_1, c_2, \ldots, c_k$.

input
```
7 2
abacaba
a b
```
output
```
12
```

input
```
10 3
sadfaasdda
f a d
```
output
```
21
```

input
```
7 1
aaaaaaa
b
```
output
```
0
```

In the first example Norge can print substrings $s[1 \ldots 2]$, $s[2 \ldots 3]$, $s[1 \ldots 3]$, $s[1 \ldots 1]$, $s[2 \ldots 2]$, $s[3 \ldots 3]$, $s[5 \ldots 6]$, $s[6 \ldots 7]$, $s[5 \ldots 7]$, $s[5 \ldots 5]$, $s[6 \ldots 6]$, $s[7 \ldots 7]$.

C. Vitamins

2 seconds, 256 megabytes

Berland shop sells n kinds of juices. Each juice has its price c_i. Each juice includes some set of vitamins in it. There are three types of vitamins: vitamin "A", vitamin "B" and vitamin "C". Each juice can contain one, two or all three types of vitamins in it.

Petya knows that he needs all three types of vitamins to stay healthy. What is the minimum total price of juices that Petya has to buy to obtain all three vitamins? Petya obtains some vitamin if he buys at least one juice containing it and drinks it.

Input

The first line contains a single integer n $(1 \leq n \leq 1\,000)$ — the number of juices.

Each of the next n lines contains an integer c_i $(1 \leq c_i \leq 100\,000)$ and a string s_i — the price of the i-th juice and the vitamins it contains. String s_i contains from 1 to 3 characters, and the only possible characters are "A", "B" and "C". It is guaranteed that each letter appears no more than once in each string s_i. The order of letters in strings s_i is arbitrary.

Output

Print -1 if there is no way to obtain all three vitamins. Otherwise print the minimum total price of juices that Petya has to buy to obtain all three vitamins.

input
```
4
5 C
6 B
16 BAC
4 A
```
output
```
15
```

input
```
2
10 AB
15 BA
```
output
```
-1
```

input

```
5
10 A
9 BC
11 CA
4 A
5 B
```

output

```
13
```

input

```
6
100 A
355 BCA
150 BC
160 AC
180 B
190 CA
```

output

```
250
```

input

```
2
5 BA
11 CB
```

output

```
16
```

In the first example Petya buys the first, the second and the fourth juice. He spends $5 + 6 + 4 = 15$ and obtains all three vitamins. He can also buy just the third juice and obtain three vitamins, but its cost is 16, which isn't optimal.

In the second example Petya can't obtain all three vitamins, as no juice contains vitamin "C".

D. Lecture Sleep

1 second, 256 megabytes

Your friend Mishka and you attend a calculus lecture. Lecture lasts n minutes. Lecturer tells a_i theorems during the i-th minute.

Mishka is really interested in calculus, though it is so hard to stay awake for all the time of lecture. You are given an array t of Mishka's behavior. If Mishka is asleep during the i-th minute of the lecture then t_i will be equal to 0, otherwise it will be equal to 1. When Mishka is awake he writes down all the theorems he is being told — a_i during the i-th minute. Otherwise he writes nothing.

You know some secret technique to keep Mishka awake for k minutes straight. However you can use it **only once**. You can start using it at the beginning of any minute between 1 and $n - k + 1$. If you use it on some minute i then Mishka will be awake during minutes j such that $j \in [i, i + k - 1]$ and will write down all the theorems lecturer tells.

You task is to calculate the maximum number of theorems Mishka will be able to write down if you use your technique **only once** to wake him up.

Input

The first line of the input contains two integer numbers n and k ($1 \le k \le n \le 10^5$) — the duration of the lecture in minutes and the number of minutes you can keep Mishka awake.

The second line of the input contains n integer numbers $a_1, a_2, \dots a_n$ ($1 \le a_i \le 10^4$) — the number of theorems lecturer tells during the i-th minute.

The third line of the input contains n integer numbers $t_1, t_2, \dots t_n$ ($0 \le t_i \le 1$) — type of Mishka's behavior at the i-th minute of the lecture.

Output

Print only one integer — the maximum number of theorems Mishka will be able to write down if you use your technique **only once** to wake him up.

input

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

output

```
16
```

In the sample case the better way is to use the secret technique at the beginning of the third minute. Then the number of theorems Mishka will be able to write down will be equal to 16.

E. Office Keys

2 seconds, 256 megabytes

There are n people and k keys on a straight line. Every person wants to get to the office which is located on the line as well. To do that, he needs to reach some point with a key, take the key and then go to the office. Once a key is taken by somebody, it couldn't be taken by anybody else.

You are to determine the minimum time needed for all n people to get to the office with keys. Assume that people move a unit distance per 1 second. If two people reach a key at the same time, only one of them can take the key. A person can pass through a point with a key without taking it.

Input

The first line contains three integers n, k and p ($1 \le n \le 1\,000$, $n \le k \le 2\,000$, $1 \le p \le 10^9$) — the number of people, the number of keys and the office location.

The second line contains n distinct integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$) — positions in which people are located initially. The positions are given in arbitrary order.

The third line contains k distinct integers $b_1, b_2, \dots, b_k$ ($1 \le b_j \le 10^9$) — positions of the keys. The positions are given in arbitrary order.

Note that there can't be more than one person or more than one key in the same point. A person and a key can be located in the same point.

Output

Print the minimum time (in seconds) needed for all n to reach the office with keys.

input

```
2 4 50
20 100
60 10 40 80
```

output

```
50
```

input

```
1 2 10
11
15 7
```

output

```
7
```

In the first example the person located at point 20 should take the key located at point 40 and go with it to the office located at point 50. He spends 30 seconds. The person located at point 100 can take the key located at point 80 and go to the office with it. He spends 50 seconds. Thus, after 50 seconds everybody is in office with keys.

F. 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
```

G. Covered Path

1 second, 256 megabytes

The on-board computer on Polycarp's car measured that the car speed at the beginning of some section of the path equals v_1 meters per second, and in the end it is v_2 meters per second. We know that this section of the route took exactly t seconds to pass.

Assuming that at each of the seconds the speed is constant, and between seconds the speed can change at most by d meters per second in absolute value (i.e., the difference in the speed of any two adjacent seconds does not exceed d in absolute value), find the maximum possible length of the path section in meters.

Input

The first line contains two integers v_1 and v_2 ($1 \le v_1, v_2 \le 100$) — the speeds in meters per second at the beginning of the segment and at the end of the segment, respectively.

The second line contains two integers t ($2 \le t \le 100$) — the time when the car moves along the segment in seconds, d ($0 \le d \le 10$) — the maximum value of the speed change between adjacent seconds.

It is guaranteed that there is a way to complete the segment so that:

- the speed in the first second equals v_1,
- the speed in the last second equals v_2,
- the absolute value of difference of speeds between any two adjacent seconds doesn't exceed d.

Output

Print the maximum possible length of the path segment in meters.

input

```
5 6
4 2
```

output

```
26
```

input

```
10 10
10 0
```

output

```
100
```

In the first sample the sequence of speeds of Polycarpus' car can look as follows: 5, 7, 8, 6. Thus, the total path is $5+7+8+6=26$ meters.

In the second sample, as $d=0$, the car covers the whole segment at constant speed $v=10$. In $t=10$ seconds it covers the distance of 100 meters.

H. Three displays

1 second, 256 megabytes

It is the middle of 2018 and Maria Stepanovna, who lives outside Krasnokamensk (a town in Zabaikalsky region), wants to rent three displays to highlight an important problem.

There are n displays placed along a road, and the i-th of them can display a text with font size s_i only. Maria Stepanovna wants to rent such three displays with indices $i < j < k$ that the font size increases if you move along the road in a particular direction. Namely, the condition $s_i < s_j < s_k$ should be held.

The rent cost is for the i-th display is c_i. Please determine the smallest cost Maria Stepanovna should pay.

Input

The first line contains a single integer n ($3 \le n \le 3\,000$) — the number of displays.

The second line contains n integers $s_1, s_2, \ldots, s_n$ ($1 \le s_i \le 10^9$) — the font sizes on the displays in the order they stand along the road.

The third line contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 10^8$) — the rent costs for each display.

Output

If there are no three displays that satisfy the criteria, print -1. Otherwise print a single integer — the minimum total rent cost of three displays with indices $i < j < k$ such that $s_i < s_j < s_k$.

input

```
5
2 4 5 4 10
40 30 20 10 40
```

output

```
90
```

input

```
3
100 101 100
2 4 5
```

output

```
-1
```

input

```
10
1 2 3 4 5 6 7 8 9 10
10 13 11 14 15 12 13 13 18 13
```

output

```
33
```

In the first example you can, for example, choose displays 1, 4 and 5, because $s_1 < s_4 < s_5$ ($2 < 4 < 10$), and the rent cost is $40 + 10 + 40 = 90$.

In the second example you can't select a valid triple of indices, so the answer is -1.

I. Mike and Fun

2 seconds, 256 megabytes

Mike and some bears are playing a game just for fun. Mike is the judge. All bears except Mike are standing in an $n \times m$ grid, there's exactly one bear in each cell. We denote the bear standing in column number j of row number i by (i, j). Mike's hands are on his ears (since he's the judge) and each bear standing in the grid has hands either on his mouth or his eyes.

They play for q rounds. In each round, Mike chooses a bear (i, j) and tells him to change his state i. e. if his hands are on his mouth, then he'll put his hands on his eyes or he'll put his hands on his mouth otherwise. After that, Mike wants to know the score of the bears.

Score of the bears is the maximum over all rows of number of consecutive bears with hands on their eyes in that row.

Since bears are lazy, Mike asked you for help. For each round, tell him the score of these bears after changing the state of a bear selected in that round.

Input

The first line of input contains three integers n, m and q ($1 \le n, m \le 500$ and $1 \le q \le 5000$).

The next n lines contain the grid description. There are m integers separated by spaces in each line. Each of these numbers is either 0 (for mouth) or 1 (for eyes).

The next q lines contain the information about the rounds. Each of them contains two integers i and j ($1 \le i \le n$ and $1 \le j \le m$), the row number and the column number of the bear changing his state.

Output

After each round, print the current score of the bears.

input

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

output

```
3
4
3
3
4
```

J. Helga Hufflepuff's Cup

2 seconds, 256 megabytes

Harry, Ron and Hermione have figured out that Helga Hufflepuff's cup is a horcrux. Through her encounter with Bellatrix Lestrange, Hermione came to know that the cup is present in Bellatrix's family vault in Gringott's Wizarding Bank.

The Wizarding bank is in the form of a tree with total n vaults where each vault has some type, denoted by a number between 1 to m. A tree is an undirected connected graph with no cycles.

The vaults with the highest security are of type k, and all vaults of type k have the highest security.

There can be at most x vaults of highest security.

Also, **if a vault is of the highest security, its adjacent vaults are guaranteed to not be of the highest security and their type is guaranteed to be less than k**.

Harry wants to consider every possibility so that he can easily find the best path to reach Bellatrix's vault. So, you have to tell him, given the tree structure of Gringotts, the number of possible ways of giving each vault a type such that the above conditions hold.

Input

The first line of input contains two space separated integers, n and m — the number of vaults and the number of different vault types possible. ($1 \le n \le 10^5, 1 \le m \le 10^9$).

Each of the next $n - 1$ lines contain two space separated integers u_i and v_i ($1 \le u_i, v_i \le n$) representing the i-th edge, which shows there is a path between the two vaults u_i and v_i. It is guaranteed that the given graph is a tree.

The last line of input contains two integers k and x ($1 \le k \le m, 1 \le x \le 10$), the type of the highest security vault and the maximum possible number of vaults of highest security.

Output

Output a single integer, the number of ways of giving each vault a type following the conditions modulo $10^9 + 7$.

input

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

output

```
1
```

input

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

output

```
13
```

input

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

output

```
0
```

In test case 1, we cannot have any vault of the highest security as its type is 1 implying that its adjacent vaults would have to have a vault type less than 1, which is not allowed. Thus, there is only one possible combination, in which all the vaults have type 2.

K. Mashmokh and ACM

1 second, 256 megabytes

Mashmokh's boss, Bimokh, didn't like Mashmokh. So he fired him. Mashmokh decided to go to university and participate in ACM instead of finding a new job. He wants to become a member of Bamokh's team. In order to join he was given some programming tasks and one week to solve them. Mashmokh is not a very experienced programmer. Actually he is not a programmer at all. So he wasn't able to solve them. That's why he asked you to help him with these tasks. One of these tasks is the following.

A sequence of l integers $b_1, b_2, ..., b_l$ $(1 \le b_1 \le b_2 \le ... \le b_l \le n)$ is called *good* if each number divides (without a remainder) by the next number in the sequence. More formally $b_i \mid b_{i+1}$ for all i $(1 \le i \le l - 1)$.

Given n and k find the number of good sequences of length k. As the answer can be rather large print it modulo 1000000007 $(10^9 + 7)$.

Input

The first line of input contains two space-separated integers n, k $(1 \le n, k \le 2000)$.

Output

Output a single integer — the number of good sequences of length k modulo 1000000007 $(10^9 + 7)$.

input

```
3 2
```

output

```
5
```

input

```
6 4
```

output

```
39
```

input

```
2 1
```

output

```
2
```

In the first sample the good sequences are: $[1, 1], [2, 2], [3, 3], [1, 2], [1, 3]$.

L. Booking System

1 second, 256 megabytes

Innovation technologies are on a victorious march around the planet. They integrate into all spheres of human activity!

A restaurant called "Dijkstra's Place" has started thinking about optimizing the booking system.

There are n booking requests received by now. Each request is characterized by two numbers: c_i and p_i — the size of the group of visitors who will come via this request and the total sum of money they will spend in the restaurant, correspondingly.

We know that for each request, all c_i people want to sit at the same table and are going to spend the whole evening in the restaurant, from the opening moment at 18:00 to the closing moment.

Unfortunately, there only are k tables in the restaurant. For each table, we know r_i — the maximum number of people who can sit at it. A table can have only people from the same group sitting at it. If you cannot find a large enough table for the whole group, then all visitors leave and naturally, pay nothing.

Your task is: given the tables and the requests, decide which requests to accept and which requests to decline so that the money paid by the happy and full visitors was maximum.

Input

The first line of the input contains integer n $(1 \le n \le 1000)$ — the number of requests from visitors. Then n lines follow. Each line contains two integers: c_i, p_i $(1 \le c_i, p_i \le 1000)$ — the size of the group of visitors who will come by the i-th request and the total sum of money they will pay when they visit the restaurant, correspondingly.

The next line contains integer k $(1 \le k \le 1000)$ — the number of tables in the restaurant. The last line contains k space-separated integers: $r_1, r_2, ..., r_k$ $(1 \le r_i \le 1000)$ — the maximum number of people that can sit at each table.

Output

In the first line print two integers: m, s — the number of accepted requests and the total money you get from these requests, correspondingly.

Then print m lines — each line must contain two space-separated integers: the number of the accepted request and the number of the table to seat people who come via this request. The requests and the tables are consecutively numbered starting from 1 in the order in which they are given in the input.

If there are multiple optimal answers, print any of them.

input
```
3
10 50
2 100
5 30
3
4 6 9
```
output
```
2 130
2 1
3 2
```

M. Journey

3 seconds❓, 256 megabytes

Recently Irina arrived to one of the most famous cities of Berland — the Berlatov city. There are n showplaces in the city, numbered from 1 to n, and some of them are connected by one-directional roads. The roads in Berlatov are designed in a way such that there **are no** cyclic routes between showplaces.

Initially Irina stands at the showplace 1, and the endpoint of her journey is the showplace n. Naturally, Irina wants to visit as much showplaces as she can during her journey. However, Irina's stay in Berlatov is limited and she can't be there for more than T time units.

Help Irina determine how many showplaces she may visit during her journey from showplace 1 to showplace n within a time not exceeding T. It is guaranteed that there is at least one route from showplace 1 to showplace n such that Irina will spend no more than T time units passing it.

Input

The first line of the input contains three integers n, m and T $(2 \le n \le 5000,\ 1 \le m \le 5000,\ 1 \le T \le 10^9)$ — the number of showplaces, the number of roads between them and the time of Irina's stay in Berlatov respectively.

The next m lines describes roads in Berlatov. i-th of them contains 3 integers u_i, v_i, t_i $(1 \le u_i, v_i \le n, u_i \ne v_i, 1 \le t_i \le 10^9)$, meaning that there is a road starting from showplace u_i and leading to showplace v_i, and Irina spends t_i time units to pass it. It is guaranteed that the roads do not form cyclic routes.

It is guaranteed, that there is at most one road between each pair of showplaces.

Output

Print the single integer k $(2 \le k \le n)$ — the maximum number of showplaces that Irina can visit during her journey from showplace 1 to showplace n within time not exceeding T, in the first line.

Print k distinct integers in the second line — indices of showplaces that Irina will visit on her route, in the order of encountering them.

If there are multiple answers, print any of them.

input
```
4 3 13
1 2 5
2 3 7
2 4 8
```
output
```
3
1 2 4
```
input
```
6 6 7
1 2 2
1 3 3
3 6 3
2 4 2
4 6 2
6 5 1
```
output
```
4
1 2 4 6
```
input
```
5 5 6
1 3 3
3 5 3
1 2 2
2 4 3
4 5 2
```
output
```
3
1 3 5
```

N. Elongated Matrix

4 seconds❓, 256 megabytes

You are given a matrix a, consisting of n rows and m columns. Each cell contains an integer in it.

You can change the order of rows arbitrarily (including leaving the initial order), but you can't change the order of cells in a row. After you pick some order of rows, you traverse the whole matrix the following way: firstly visit all cells of the first column from the top row to the bottom one, then the same for the second column and so on. During the traversal you write down the sequence of the numbers on the cells in the same order you visited them. Let that sequence be $s_1, s_2, \dots, s_{nm}$.

The traversal is k-acceptable if for all i $(1 \le i \le nm - 1)$ $|s_i - s_{i+1}| \ge k$.

Find the maximum integer k such that there exists some order of rows of matrix a that it produces a k-acceptable traversal.

Input

The first line contains two integers n and m $(1 \le n \le 16,$ $1 \le m \le 10^4,$ $2 \le nm)$ — the number of rows and the number of columns, respectively.

Each of the next n lines contains m integers $(1 \le a_{i,j} \le 10^9)$ — the description of the matrix.

Output

Print a single integer k — the maximum number such that there exists some order of rows of matrix a that it produces an k-acceptable traversal.

input

```
4 2
9 9
10 8
5 3
4 3
```

output

```
5
```

input

```
2 4
1 2 3 4
10 3 7 3
```

output

```
0
```

input

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

output

```
3
```

In the first example you can rearrange rows as following to get the 5-acceptable traversal:

```
5 3
10 8
4 3
9 9
```

Then the sequence s will be $[5, 10, 4, 9, 3, 8, 3, 9]$. Each pair of neighbouring elements have at least $k = 5$ difference between them.

In the second example the maximum $k = 0$, any order is 0-acceptable.

In the third example the given order is already 3-acceptable, you can leave it as it is.

O. Hard Process

1 second, 256 megabytes

You are given an array a with n elements. Each element of a is either 0 or 1.

Let's denote the length of the longest subsegment of consecutive elements in a, consisting of only numbers one, as $f(a)$. You can change no more than k zeroes to ones to maximize $f(a)$.

Input

The first line contains two integers n and k $(1 \le n \le 3 \cdot 10^5, 0 \le k \le n)$ — the number of elements in a and the parameter k.

The second line contains n integers a_i $(0 \le a_i \le 1)$ — the elements of a.

Output

On the first line print a non-negative integer z — the maximal value of $f(a)$ after no more than k changes of zeroes to ones.

On the second line print n integers a_j — the elements of the array a after the changes.

If there are multiple answers, you can print any one of them.

input

```
7 1
1 0 0 1 1 0 1
```

output

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

input

```
10 2
1 0 0 1 0 1 0 1 0 1
```

output

```
5
1 0 0 1 1 1 1 1 0 1
```