COCI 2017/2018

Round #7, March 3rd, 2018

Tasks

Task	Time limit	Memory limit	Score
Prosjek	1 s	64 MB	50
Timovi	1 s	64 MB	80
Go	1 s	512 MB	100
Clickbait	1 s	128 MB	120
Dostavljač	2 s	64 MB	140
Priglavci	2 s	64 MB	160
Total			650

Little Ivica received *N* math grades and wants to calculate their average. He knows that the average of two numbers *a* and *b* is calculated as (*a* + *b*) / 2, but he still doesn't know how to do it for multiple numbers. He calculates the average by writing down *N* grades and repeating the following operations *N* - 1 times:

1. He chooses two numbers and erases them.
2. He writes down the average of the two chosen numbers.

After precisely *N* - 1 steps, the only number written down will be the one representing the average grade to Ivica. It is your task to determine the largest average that can be obtained this way.

INPUT

The first line of input contains the integer N ($1 \le N \le 20$), the number from the task.
The i^{th} of the following N lines contains the integer X_i ($1 \le X_i \le 5$), the i^{th} grade.

OUTPUT

Output the largest possible average from the task. Your solution is allowed to deviate from the official one for 0.000001.

SCORING

In test cases worth 20% of total points, it will hold N = 3.
In test cases worth an additional 20% of total points, it will hold N = 4.
In test cases worth an additional 20% of total points, it will hold N = 5.

SAMPLE TESTS

input

```
4
2
4
5
2
```

output

```
4.000000
```

input

```
3
5
5
4
```

output

```
4.750000
```

input

```
3
1
3
5
```

output

```
3.500000
```

Clarification of the third test case:
Initially, the numbers 1, 3 and 5 are written down.
In the first step, Ivica chooses numbers 1 and 3, erases them and writes down 2. After the first step, 2 and 5 are written down.

In the second step, Ivica chooses the remaining two numbers which average is 3.5.

We need to arrange M kids in N teams. We begin by placing K kids in each team, starting from the first to the N^{th} team. When we finish with the N^{th} team, we switch directions and continue, placing K kids in each team, from the $(N\text{-}1)^{\text{th}}$ to the first team, respectively. When we finish with the first team, we switch directions again and continue the process from the second to the N^{th} team, respectively, and so on until there are no kids left to distribute. For example, if we have three teams, we will place K kids in teams in the following order: first team, second team, third team, second team, first team, second team, etc.

If, at any points, there are less than K kids left to place in the current team, we place all the remaining kids in that team and end the process.
Output the number of kids in each team when the process ends.

INPUT

The first line of input contains the integers N ($2 \le N \le 200\ 000$), K and M ($1 \le K \le M \le 2\ 000\ 000\ 000$) from the task.

OUTPUT

In a single line, output the number of kids in each of the N teams, respectively from the first to the N^{th} team.

SCORING

In test cases worth a total of 40 points, it will hold $M / K \le 200\ 000$.

SAMPLE TESTS

input
```
2 1 3
```
output
```
2 1
```

input
```
3 2 7
```
output
```
2 3 2
```

input
```
4 5 6
```
output
```
5 1 0 0
```

Branimirko is still a passionate player of the world-renowned game Pokémon Go. Recently, he decided to organize a competition in catching Pokémon. It will be held in Ilica street in Zagreb, and the main sponsor is his friend Slavko. The reward is, of course, candy!

Ilica is, as we all know, the longest street in Zagreb. There are N houses on the same side of the street, and each house has a house number between 1 and N. The competition begins at house number K.

Before the competition, Branimirko looked at the map and saw M Pokémon. Each Pokémon is located at its (distinct) house number A_i, is valued at B_i candy, and can be caught only in the next T_i seconds, after which it disappears from the map and is impossible to catch.

Branimirko can visit one house number per second. Also, when he catches a Pokémon, that Pokémon disappears from the map.

Since Branimirko really likes candy, he asked for your help.
Help him and determine the maximal amount of candy he can get!

INPUT

The first line of input contains integers N, K ($1 \le K \le N \le 1\ 000$) and M ($1 \le M \le 100$), the number of houses, the starting house number and the number of Pokémon.
Each of the following M lines contains integers A_i ($1 \le A_i \le N$), B_i ($1 \le B_i \le 100$) and T_i ($1 \le T_i \le 2\ 000$) from the task.
In the input data, the Pokémon will always be in a strictly ascending order by house number A_i.

OUTPUT

You must output the required maximal amount of candy from the task.

SCORING

In test cases worth 20% of total points, it will hold $M \le 10$.
In additional 20% of total points, it will hold $M \le 20$.

SAMPLE TESTS

input

```
10 5 4
1 30 4
3 5 7
7 10 12
9 100 23
```

output

```
115
```

input

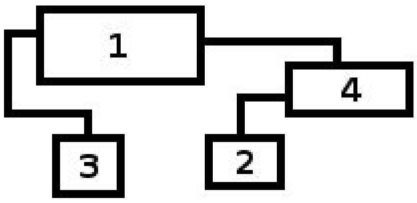
```
20 8 7
1 35 14
4 57 1
6 32 2
9 94 28
14 78 8
15 8 1
17 55 3
```

output

```
172
```

Clarification of the first test case:
One strategy is that Branimirko first catches the Pokémon at house number 3 (5 candy), then, respectively, house numbers 7 (10 candy) and 9 (100 candy) for a total of 115 candy.
Notice that Branimirko can't catch the Pokémon at house number 1, because he can't reach it in time from his starting position (house number 5).

While surfing online, Slavko came across an ad displaying a system of containers and pipes (an example of such system is illustrated in the image below) with the message: “If container **1** starts filling up with water, determine the order in which the containers get filled up.”



The system consists of *K* containers denoted with numbers from 1 to *K*, and we can describe it using a matrix of characters with *N* rows and *M* columns. The containers are **in the shape of a rectangle**, and the outlines of the containers and pipes are shown with the following characters:

- ‘-’ if it’s a horizontal part of the outline,
- ‘|’ if it’s a vertical part of the outline, and
- ‘+’ if it’s a spot where the horizontal and vertical parts of the outline connect. An exception is where the containers and pipes connect. In that case, the container outline dominates (see sample tests).

In an arbitrary place within each container, there is a string of digits that represent the label of the container, and all the other fields in the matrix are equal to ‘.’ (dot).

All containers except the one labeled with 1 have **exactly one** supply pipe that enters the container in its **upper side**. The container labeled with 1 does not have a supply pipe.

The containers can have multiple (also possible, zero) discharge pipes that leave the container out of its **lateral side**. The places where discharge pipes leave a container will be in mutually **distinct rows** in the matrix.

The pipes directly connect two containers, which means that **it is not possible** to split the pipes or connect multiple pipes into one, and no two pipes will intersect. On their way, looking from the source to the destination container, the pipes always descend to the following row or stay in the same row. In other words, they never go back to the previous row, so the water can flow freely from one container to another.

The water enters a container until it is full. If the water level reaches the level of the discharge pipe, the water will flow through that pipe until the container the pipe leads into is filled up.

Help Slavko and determine the order in which the containers will fill up.

Please note:

- The test data is such that each character ‘+’ is surrounded with **exactly one character** ‘-’ to the left or the right side and **exactly one** character ‘|’ to the upper or lower side, and **all other** adjacent characters in directions up, down, left and right will be equal to ‘.’ (dot).

- The only places where the pipe in the matrix is in a field adjacent to the container outline are the places where the pipe enters or exits the container. In other words, a pipe will never run right next to a container (except where it connects with the container). The entry for the supply pipe is labeled with the character ‘|’ above a container, whereas the exit of the discharge pipe is labeled with the character ‘-’ on the lateral side of a container.

INPUT

The first line contains two integers *N* and *M* ($1 \le N, M \le 1000$), matrix dimensions.
The following *N* lines contain *M* characters describing the container system.

OUTPUT

You must output *K* lines. The i^{th} line contains the label of the container that fills up i^{th}. A solution will always exist and will be unique.

SCORING

In test cases worth 70% of total points, it will hold $N, M \le 100$.

SAMPLE TESTS

input

```
12 13
..+--+.......
+-|..|.......
|.|.1|--+....
|.+--+..|....
|......+----+
+---+..|..2.|
....|..+----+
.+--+........
.|...........
+---+........
|.3.|........
+---+........
```

output

```
2
3
1
```

input

```
8 10
..........
.......+-+
...+---|1|
...|...+-+
...|......
..+-+.....
..|2|.....
..+-+.....
```

output

```
2
1
```

Clarification of the first test case:
Container 1 starts filling up with water.
The water level in container 1 grows, and in one moment reaches the level of the discharge pipe leading to container 2. The water flows through the pipe until container 2 fills up.
After that, the water level in container 1 keeps growing until it reaches the level of the discharge pipe leading to container 3, which fills up next.
Finally, the water level in container 1 keeps growing and the container fills up.

Ever since Krešo started growing chili peppers, N restaurants all over Croatia showed interest in his peppers so they could enrich their dishes with real spice. Because of high demand, Krešo decided to start working as a delivery guy of chili peppers.

The restaurants are denoted with numbers from 1 to N and are mutually connected with N - 1 roads such that traveling between any two restaurants is possible. Krešo begins his journey in the restaurant denoted with 1. In one unit of time, he can either drive to the adjacent restaurant or deliver the peppers to the current restaurant. For each restaurant, we know the required amount of peppers A_i.

Since delivering goods is tiring, Krešo decided to spend a total of M units of time on delivery and travel, after which he plans to take a break.

Determine the maximal amount of peppers Krešo can deliver in the given timeframe. You can assume that Krešo always carries an unlimited supply of peppers.

INPUT

The first line of input contains two integers N and M ($1 \le N, M \le 500$), the number of restaurants and the time Krešo plans to spend delivering peppers.
The second line of input contains N integers A_i ($1 \le A_i \le 10^6$), the required amount of peppers for restaurants denoted with i ($1 \le i \le N$).
Each of the following N - 1 lines contains two integers U and V ($1 \le U, V \le N, U \ne V$) that represent the road between restaurants U and V.

OUTPUT

You must output the maximal amount of peppers Krešo can deliver in the given timeframe.

SAMPLE TESTS

input

```
3 5
9 2 5
1 2
1 3
```

output

```
14
```

input

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

output

```
3
```

input

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

output

```
15
```

Clarification of the first test case:
In the first step, Krešo will deliver the peppers to restaurant 1.
In the second step, Krešo will travel to restaurant 3.
In the third step, Krešo will deliver the peppers to restaurant 3.
He is left with 2 units of time, in which he can drive to restaurant 2, but he lacks one unit of time to deliver the peppers to that restaurant.

Engineer Zlatko got assigned the task of checking the quality of transportation for students getting to school by bus. In a 2D-coordinate system, there are *N* students with coordinates u_x and u_y, and *M* bus stops with coordinates s_x and s_y. At the beginning, a field can be occupied by one student or by one stop, or it can be empty.

Also, engineer Zlatko has a list of *K* bus lines: for each bus line, he has a list of stops the bus stops at in the order in which the stops are listed. One stop belongs exclusively to one bus line. The stops are distinct within a bus line. There is only one bus per line. Additionally, each bus can fit *C* students. Bus stops don't have a limit on the number of students that could be waiting for it.

When a student boards a bus, they don't get off until the end of the ride when the bus has stopped at all stops of the line. A student can board only one bus. For a student to board a bus, they must reach a stop of one of the bus lines. The **length of the path** which a student walked from their position to a bus stop is measured as the **squared** Euclidean distance: $(u_x - s_x)^2 + (u_y - s_y)^2$.

Engineer Zlatko chooses the boarding stop for each student and distributes them so that the buses can fit everyone, respecting the given limitations. The **weakness** of the distribution of students is measured as the distance walked by the student farthest from their boarding bus stop.

Help engineer Zlatko and calculate the **minimal** possible weakness and the distribution of students.

INPUT

The first line of input contains integers *N, M, C, K* ($1 \le N, M, C, K \le 100$) from the task.
Each of the following *N* lines contains integers u_x and u_y ($-1000 \le u_x, u_y \le 1000$), the students' coordinates.
Each of the following *M* lines contains integers s_x and s_y ($-1000 \le s_x, s_y \le 1000$), the stops' coordinates.
Each of the following *K* lines contains the list of bus stops: first, the number of stops K_i of the bus line, then K_i numbers st_j ($1 \le st_j \le M$) that denote the stops.

OUTPUT

If it is possible to distribute the students within the requirements, you must output the required weakness in the first line, and in the following *N* lines, the *i*th line must contain the stop the *i*th student must walk to. In the case that the distribution of students to bus stops with the calculated weakness is not unique, output an arbitrary distribution with such weakness.
If it is impossible to distribute the students, you must output '-1' (without quotes).

SCORING

In test cases worth 50% of total points, it will hold *C = 1*.
In test cases worth additional 30% of total points, it will hold *1 ≤ C ≤ 10.*

SAMPLE TESTS

input

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

output

```
4
1
1
```

input

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

output

```
-1
```

input

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

output

```
9
1
1
3
```

Clarification of the first test case:
The distance which both students must walk to a bus stop is 2. The squared value of that instance is 4.
Clarification of the second test case:
Since only one line exists, in total there is a single bus with a capacity of 1, which is not sufficient to distribute all the students properly.
Clarification of the third test case:
Firstly, two students go to the first stop. The nearest stop to the third student is the second stop, but that stop belongs to the bus line of an already full bus. Therefore, the third student must go to the third stop, and the squared value of their path length is 9. Every other distribution results in greater weakness.