CODECON

Memory Limit: 1024 MB
Time Limit: 1 s

A Very Odd Problem (100 points)

Introduction

As part of his religion, your friend hates odd numbers and empty arrays. He hates odd numbers and empty arrays so much that whenever he encounters an array, he tries to remove some values such that the resulting array has an even sum. Furthermore, the resulting array must contain a non zero even amount of even numbers after the transformation. When your friend can not remove any values from the array such that the array satisfies the conditions given before, he becomes insane. Being a good friend, you decide to check whether it is safe to give your friend an array.

Input Specifications

The first line contains n (1 <= n <= 10'000), the number of values in the array. The next line contains 'n' space separated integers, the n elements of the array. Every value is contained in the closed interval [0; 999'999].

Output Specifications

Print "YES" if it is safe to give your friend the array. Otherwise, print "NO".

Sample Input/Output

Input

```
5
4 4 5 8 1
```

Output

```
YES
```

Explanation

```
Your friend can remove all values except 4 and 4. Alternatively, he could have also
kept values 4 and 8 or the other 4 and 8.
```

Input

```
4
1 2 3 5
```

Output

```
NO
```

Explanation

```
No matter which elements your friend removes, he will never have a non zero even
number of even values remaining.
```

Memory Limit: 1024 MB
Time Limit: 5 s

Tracing Vile Angus (200 points)

Introduction

Inspector Catchburgle is on a case! He is chasing a mysterious villain that goes by the name Vile Angus. Fortunately, Angus has a bad memory and tends to write a lot of things down. And Catchburgle just found a pile of the scoundrel's notes! There must be lots of valuable clues in them!

However, Angus isn't stupid; he was encrypting his messages with some cipher, so that they are not so easy to read.

Inspector Catchburgle has applied his brilliant logic, and found out that the villain uses some sort of a substitution cipher. It seems that he goes over the text multiple times, and each time swaps two letters in part of the text from a random position to the end.

But that's not all! The bright detective has even figured out the mechanism by which Angus decides which letters to swap and from which position.

Well, inspector Catchburgle didn't share the details of this mechanism with you, but before he went on a well deserved coffee break, he wrote down all the substitutions for each encoded message.

Will you, the inspector's assistant, be able to impress the inspector by decoding the messages before he returns from his coffee break? It will surely help to catch the villain (and you will deserve part of the credit)!

Input Specifications

- First line of input is the message as Vile Angus wrote it. Its length is in the interval *0 ≤ length ≤ 10000*. (Yes, there were some blank papers, too. The inspector didn't bother removing them.)
- Second line is number *N*, where *0 ≤ N ≤ 1000*.
- The next *N* lines each contains information about a pair of letters that the inspector found to be swapped. Each line is in the format `pos first second`, and means that starting at position *pos* (zero-based), each occurrence of letter *first* has been replaced by letter *second* and vice versa. The values of *pos* form a non-decreasing sequence.

Note that:

- There can be multiple swapped pairs starting at the same position.
- The order of swaps is important. When Angus encrypted his messages, he was applying the substitutions in the order in which you receive them on input.
- The letters *first* and *second* are always lower case letters, but when doing the substitutions, Angus keeps the upper- or lower-case of the original letter. So if the original letter was upper case, the substituted letter will also be upper case.
- Some positions *pos* may be past the end of the message. Inspector Catchburgle might have gotten carried away a little...

Output Specifications

Print the villain's message with all the letter swaps undone.

Sample Input/Output

Input

```
Vile Angus
0
```

Output

```
Vile Angus
```

Explanation

There are 0 substitutions. The text is not encrypted at all.

Input

```
Diury uf Vise Ongal
3
0 a u
6 l s
6 o u
```

Output

```
Diary of Vile Angus
```

Explanation

When Angus was encrypting the message, he did 3 passes:

```
Diary of Vile Angus
```

(from beginning of the text onward, a is swapped with u)

```
Diury of Vile Ungas
```

(from beginning of second word onward, l is swapped with s)

```
Diury of Vise Ungal
```

(from beginning of second word onward again, o is swapped with u)

```
Diury uf Vise Ongal
```

Input

```
The password to my safe is: f1ddler
5
28 k f
28 t d
28 l e
28 n e
28 s r
```

Output

```
The password to my safe is: k1ttens
```

Explanation

- There are no substitutions until the last word (position 28). From there, *f* is replaced with *k* ("k1ttens" -> "f1ttens").
- Also, *t* is replaced with *d* ("f1ttens" -> "f1ddens").
- The order of the following two swaps, *l* <-> *e* and *n* <-> *e*, is important ("f1ddens" -> "f1ddlns" -> "f1ddles").
- Finally, *s* is swapped with *r* ("f1ddles" -> "f1ddler").

Input

```
Catdhburgfe has no icea vhav I hate buqiec vhe bocy in vhe gaqcen.
6
2 c d
9 f l
15 v t
16 s d
17 r q
31 h h
```

Output

```
Catchburgle has no idea that I have buried the body in the garden.
```

Input

```
Trkin to Viennk: Juexdky, 7:15km, 3rd ajkas
12
2 a k
2 q s
4 q x
6 h s
8 t j
14 s s
16 a c
19 o j
23 f v
26 z o
31 x f
42 j y
```

Output

```
Train to Vienna: Tuesday, 7:15am, 3rd coach
```

Explanation

Swapping *s* for *s* from position 14 is actually a no-op. Also, some of the swaps towards the end do not actually swap anything because the letters don't appear in the respective part of the text.

Input

```
Angus, on your way fpoa wope, rbkmdk luy ahbe, lpkmg mng rotmtokd. Timned, Kbh.
7
```

```
7 b l
15 h i
15 e k
17 a m
18 p r
22 s g
34 d g
```

Output

```
Angus, on your way from work, please buy milk, bread and potatoes. Thanks, Eli.
```

Input

```
People that I have afbushed and mobbed ro wam: Heimy C., Rngflid E., Evaigeunia de
M., Aetem A. (tince), pfd thpt gly inth the wlffy soldtpche
11
3 w r
15 m w
18 w f
25 r s
32 u l
33 i n
82 a p
85 s i
98 f s
109 f f
127 d r
```

Output

```
People that I have ambushed and robbed so far: Henry C., Sigmund E., Evangelina de
R., Peter P. (twice), and that guy with the funny moustache
```

Input

```
2
0 n o
0 o n
```

Output

Explanation

This text is empty, there is nothing to swap. The output is empty as well.

Memory Limit:	1024 MB
Time Limit:	5 s

Passport Control (400 points)

Introduction

Upon arrival at an international airport, passengers must go through passport control. They are arranged in a line and must go to the lowest numbered available passport control booth, where a border agent will check their travel documents.

Passengers can come alone or in groups. You should assume that every passenger will get through passport control after 1 minute, so a group of 5 passengers will have all its checks completed after 5 minutes. Each group of passengers will be processed by a single border agent.

Each passenger group is immediately assigned to the lowest numbered available border agent, so you should assume no time is spent for this.

Case 1: The group of passengers should go to Booth 2

```
                    Booth 1      Booth 2     Booth 3     ...   Booth N
Passengers ---->    Unavailable  Available   Available   ...   Available
```

Case 2: The group of passengers should go to Booth 1

```
                    Booth 1    Booth 2     Booth 3       ...   Booth N
Passengers ---->    Available  Available   Unavailable   ...   Available
```

After checking 10 passenger groups (regardless of size), each border agent is allowed to take a break, which will make their respective booth unavailable during 5 minutes.

Your task is to count how many groups were through each booth.

Input Specifications

- $1 \leq N \leq 10^4$
- $1 \leq M \leq 10^6$
- $1 \leq groupSizes \leq 10^2$

The first line (N) will be the number of available booths at passport control. The second line (M) will contain how many groups are arriving. The next M lines will contain the number of passengers in each group.

Output Specifications

You should output a space separated list of integers containing how many groups each border agent has processed. Note that the sum of these integers must be equal to M.

Sample Input/Output

Input

```
3
6
4
```

```
2
1
3
5
1
```

Output

```
2 2 2
```

Explanation

Agents 1, 2 and 3 will take groups with 4, 2 and 1 passengers at first, respectively. Then, agent 3 will take a group with 3 passengers, agent 2 will process the group with 5 and finally agent 3 will let through the final passenger. Each agent will have processed 2 groups.

Input

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

Output

```
1 2 3
```

Explanation

Agents 1, 2 and 3 will take groups with 4, 2 and 1 passengers at first, respectively. Then, agent 3 will take a group with 2 passengers, agent 2 will process the group with 5 and finally agent 2 will let through the final passenger. The final count will be 1, 2 and 3.

Input

```
10
19
1
10
10
10
10
10
10
10
10
10
1
1
1
```

```
1
1
1
1
1
1
```

Output

```
10 1 1 1 1 1 1 1 1 1
```

Explanation

Bad luck, Agent #1

Input

```
10
20
1
10
10
10
10
10
10
10
10
10
1
1
1
1
1
1
1
1
1
1
```

Output

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

Explanation

Take a rest, Agent #1

Input

```
10
19
10
10
```

```
10
10
10
10
10
10
10
1
1
1
1
1
1
1
1
1
1
```

Output

```
1 1 1 1 1 1 1 1 1 10
```

Explanation

Bad luck, Agent #10

Input

```
2
17
1
10
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
```

Output

```
11 6
```

Explanation

Agent #1 returns from break

Memory Limit: 1024 MB
Time Limit: 2 s

Goodie Collector (300 points)

Introduction

The big day has finally arrived, Bloomberg has come to your university and prepared a fun contest for you. To satisfy basic student needs, Bloomberg has prepared some tables arranged in a row with some different goodies including socks, pens, and other useful items. Due to logistic reasons, there is only a single type of goodie on each table. Two adjacent tables are exactly one metre apart, every table is considered of negligible width for this problem and the tables are numbered from left to right. Of course, you want to maximize the number of different goodies you can get. One restriction applies however: as you do not want to appear greedy to your fellow friends, you want to make sure that between every two tables you take goodies from there is at least some given distance.

Input Specifications

The first line contains the integers n, k and d that are separated by a space. The integer n is the number of tables (1 <= n <= 12), k is the number of different goodies (1 <= k <= 12) and d is the minimum distance in meters between two tables you take goodies from (1 <= d <= n).

The second line contains n space separated integers. The i-th integer is the goodie that can be picked up on the i-th table. A goodie type is a number between 0 (inclusive) and k (exclusive).

Output Specifications

Print the maximum number of different goodie types you can get.

Sample Input/Output

Input

```
8 3 3
1 2 2 1 0 0 0 0
```

Output

```
2
```

Explanation

```
Here, one can only select two different types of goodies.
Either select goodie types 0 and 1 by going to the tables (which are assumed to be
1-indexed):
- 1 4 7
- or 1 4 8
- or 1 5 8
- or 1 5
- or 1 6
- or 1 7
- or 1 8
- or 4 7
```

- or 4 8
Or select goodie types 0 and 2 by going to the the tables:
- 2 5 8
- or 2 5
- or 2 6
- or 2 7
- or 2 8
- or 3 6
- or 3 7
- or 3

Input

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

Output

```
3
```

Explanation

It is possible to get 3 different types of goodies from tables 1, 3 and 5. This is optimal as goodie type 1 is not available anywhere and this is the only goodie type we do not get.

CODECON

Memory Limit: 1024 MB
Time Limit: 6 s

Multiplayer Game (600 points)

Introduction

We have two players playing a game on a square grid that contains rewards.
The two players start from the top left cell of the grid and on each move each of them moves down or to the right.
They move at the same time and need to collaborate to collect the most number of rewards.
They can occupy the same cell in the grid at the same time but the reward, if any exists in the cell, is collected only once.
The game finishes when there are no more possible moves.

Input Specifications

The first line of the input contains a single integer N - the size of one side of the square grid.
N lines follow describing a row of the game grid each.

Each line contains N digits (either 0 or 1) describing the cells in that row. 0 - meaning an empty cell, and 1 - meaning a cell with reward. There is no whitespace between the digits.

1 <= N <= 100

Output Specifications

Output a single integer - the maximum score that the players can achieve playing together.

Sample Input/Output

Input

```
4
0101
1000
0010
1000
```

Output

```
4
```

Explanation

No matter how the two players move, they can not collect all of the rewards.

CODECON

Memory Limit: 1024 MB
Time Limit: 5 s

Warrior Of Hogwarts (500 points)

Introduction

Henry is a curious student, studying at Hogwarts. Being smarter, faster and displaying more zeal for magic than other students, he became popular among three witches at the school. They knew Henry has a secret desire to be a warrior. So each gave him a super power to fight against his enemies.

`The first witch gave Henry the power to reduce the strength of his enemies by 1 unit.`

`The second witch gave Henry the power to divide the strength of his enemies by 2.`

`The third witch gave Henry the power to divide the strength of his enemies by 3.`

The witches clearly told him that the strength of every enemy must remain an integer at all times.

Henry can defeat an enemy when the strength of the enemy equals 1.

K is the initial strength of his enemy. Help Henry to find the minimum number of times he needs to apply his super powers to defeat the enemy and become a warrior.

Input Specifications

The input consists of the integer **N**, the initial strength of Henry's enemy.

Constraints

```
1 <= N <= 1'000'000'000
```

Output Specifications

Print the minimum number of hits needed to defeat his enemy by making his strength equal to 1.

Sample Input/Output

Input

```
5
```

Output

```
3
```

Explanation

Input = 5, Reduce by 1 and then reduce power by half 2 times **Answer=3**

Input

```
1
```

Output

```
0
```

Explanation

Input = 1, No need to to any operations **Answer=0**

Input

```
2
```

Output

```
1
```

Explanation

Input = 2, Reduce By half **Answer=1**

CODECON

Memory Limit: 1024 MB
Time Limit: 5 s

Mother of Dragons (500 points)

Introduction

Daenerys Targaryen can rule over 7 kingdoms if she has all her dragons by her side.

Daenerys has built **N** dragon-pits in her kingdom. The dragon-pits are located in a straight line at positions **x1, x2, …, xN.**

She has **C** dragons. The dragons are feisty and don’t like being restrained. Once they are put into the dragon-pits they try to burn and bite each other.

Tyrion, being her loyal and intelligent adviser, is given the task of preventing the dragons from hurting each other. To do so Tyrion wants to assign the dragons such that minimum distance between the dragons is as large as possible.

Help Tyrion find the largest minimum distance.

Input Specifications

- Line 1: Two space-separated integers: N and C

  N ranges as follows **(C <= N <= 20)**
  C ranges such that **(2 <= C <= N)**

- Lines 2..N+1: Line i+1 contains an integer which is the position of the dragon-pit (xi) **x1, x2, …, xN (0 <= xi <= 1'000'000'000)**

The position of the pits can be given in any order (not necessarily in sorted order)

Output Specifications

Print the largest possible minimum distance between the dragons.

Sample Input/Output

Input

```
5  3
1
2
8
4
9
```

Output

```
3
```

Explanation

Tyrion can put 3 Dragons in the Dragon-pit at positions 1, 4 and 8,

resulting in a minimum distance of 3.

Input

```
4 2
100000000
500000000
900000000
1
```

Output

```
899999999
```

CODECON

Memory Limit: 1024 MB
Time Limit: 5 s

Forest Party (900 points)

Introduction

Together with your friends, you organize a party in the forest. What would a party be without someone making a fool of themselves? To help loosen up the atmosphere and make the evening fun, you decide to create a little game. In this game, everyone has to travel between two places taking up the pre-set challenges as they come, (which were cunningly prepared by you the previous day). For instance, you may have to cross a small river over a fallen tree trunk, or you may have to walk some distance with stilts. To reach the target destination, you might have to use several paths. For every path that you use, you must take up the challenge along that path.

As you are an absolutely evil person, your goal is not to finish the race first, but to make the others look more foolish than you. To achieve this, you categorized every path by its potential to make a fool of yourself and gave every path a FI (Foolishness Index) number. A higher number means that you will very likely make a fool of yourself. Your goal is now to find how to reach the destination by keeping the maximum FI number that you encounter as low as possible. The FI number for a sequence of paths is the maximum FI number of a path of that sequence.

Unfortunately, you do not know yet where the game will start and where it will end. As you suspect that some configurations are more likely than others (your even more evil friend wants you to fall into the river), you prepared a list of very likely routes. Every route contains the start location and the end location. For every route, you want to find out the minimum FI number over all sequences of paths between the starting location and the ending location.

An empty sequence of paths is considered to have an FI number of 0. Furthermore, there may be several direct paths between the same locations. There might even be direct paths that loop back to their start location. All paths can be used in both directions. Finally, at the time that you created the paths and the corresponding challenges, you made sure that it is always possible to reach every location from every location.

Input Specifications

The first line contains two space separated integers n and m. Integer n is the number of locations in the forest, and m is the number of paths in the forest. There are at most 100,000 locations and at most 100,000 paths.

The next m lines contain three space separated integers that represent one path. The first two integers of a path are the two locations that the path links together. Path identifiers are integers between 1 and n. The last integer that describes a path is the FI number, which is at least 0 and at most 1,000,000,000.

The next lines contain the number of queries you want to answer. The number of queries is at most 10,000.

Each of the next q lines contains a query. Every query contains two space separated path identifiers.

Output Specifications

For every query, output the minimum FI number of a sequence of paths that links these two locations together followed by a newline character.

Sample Input/Output

Input

```
8 11
7 1 8
4 8 4
6 8 0
5 3 0
2 2 7
3 2 5
1 3 3
1 6 4
5 4 6
1 6 5
6 2 1
2
3 4
8 8
```

Output

```
4
0
```

Explanation

For the first query, the sequence of paths that minimizes the FI number between locations 3 and 4 is 3, 1, 6, 8, 4. For the second query, we are already at the ending location, thus the best FI number we can get is 0.