Lista AA 2022 #2

A. The least round way

2 seconds, 64 megabytes

There is a square matrix $n \times n$, consisting of non-negative integer numbers. You should find such a way on it that

- starts in the upper left cell of the matrix;
- each following cell is to the right or down from the current cell;
- the way ends in the bottom right cell.

Moreover, if we multiply together all the numbers along the way, the result should be the least "round". In other words, it should end in the least possible number of zeros.

Input

The first line contains an integer number n $(2 \le n \le 1000)$, n is the size of the matrix. Then follow n lines containing the matrix elements (non-negative integer numbers not exceeding 10^9).

Output

In the first line print the least number of trailing zeros. In the second line print the correspondent way itself.

input
```
3
1 2 3
4 5 6
7 8 9
```
output
```
0
DDRR
```

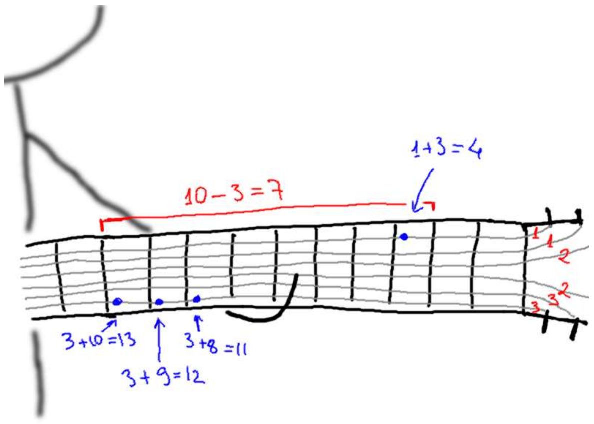
B. Perform Easily

2 seconds, 256 megabytes

After battling Shikamaru, Tayuya decided that her flute is too predictable, and replaced it with a guitar. The guitar has 6 strings and an infinite number of frets numbered from 1. Fretting the fret number j on the i-th string produces the note $a_i + j$.

Tayuya wants to play a melody of n notes. Each note can be played on different string-fret combination. The easiness of performance depends on the difference between the maximal and the minimal indices of used frets. The less this difference is, the easier it is to perform the technique. Please determine the minimal possible difference.

For example, if $a = [1, 1, 2, 2, 3, 3]$, and the sequence of notes is $4, 11, 11, 12, 12, 13, 13$ (corresponding to the second example), we can play the first note on the first string, and all the other notes on the sixth string. Then the maximal fret will be 10, the minimal one will be 3, and the answer is $10 - 3 = 7$, as shown on the picture.



Input

The first line contains 6 space-separated numbers a_1, a_2, ..., a_6 ($1 \le a_i \le 10^9$) which describe the Tayuya's strings.

The second line contains the only integer n $(1 \le n \le 100\,000)$ standing for the number of notes in the melody.

The third line consists of n integers b_1, b_2, ..., b_n $(1 \le b_i \le 10^9)$, separated by space. They describe the notes to be played. It's guaranteed that $b_i > a_j$ for all $1 \le i \le n$ and $1 \le j \le 6$, in other words, you can play each note on any string.

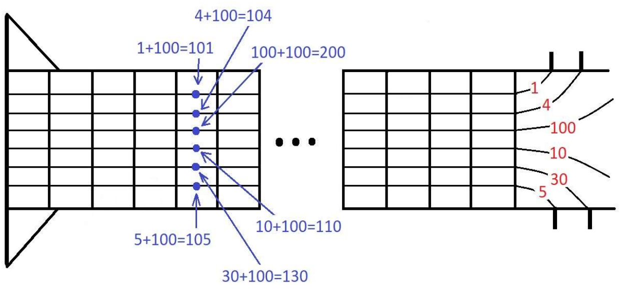
Output

Print the minimal possible difference of the maximal and the minimal indices of used frets.

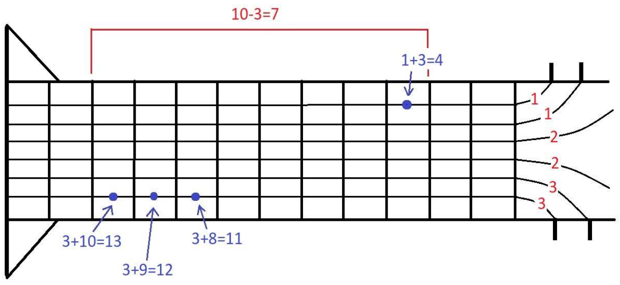
input
```
1 4 100 10 30 5
6
101 104 105 110 130 200
```
output
```
0
```

input
```
1 1 2 2 3 3
7
13 4 11 12 11 13 12
```
output
```
7
```

In the first sample test it is optimal to play the first note on the first string, the second note on the second string, the third note on the sixth string, the fourth note on the fourth string, the fifth note on the fifth string, and the sixth note on the third string. In this case the 100-th fret is used each time, so the difference is $100 - 100 = 0$.



In the second test it's optimal, for example, to play the second note on the first string, and all the other notes on the sixth string. Then the maximal fret will be 10, the minimal one will be 3, and the answer is $10 - 3 = 7$.



C. Busy Robot

2 seconds, 256 megabytes

You have a robot that can move along a number line. At time moment 0 it stands at point 0.

You give n commands to the robot: at time t_i seconds you command the robot to go to point x_i. Whenever the robot receives a command, it starts moving towards the point x_i with the speed of 1 unit per second, and he stops when he reaches that point. However, while the robot is moving, it **ignores** all the other commands that you give him.

For example, suppose you give three commands to the robot: at time 1 move to point 5, at time 3 move to point 0 and at time 6 move to point 4. Then the robot stands at 0 until time 1, then starts moving towards 5, ignores the second command, reaches 5 at time 6 and immediately starts moving to 4 to execute the third command. At time 7 it reaches 4 and stops there.

You call the command i successful, if there is a time moment in the range $[t_i, t_{i+1}]$ (i. e. after you give this command and before you give another one, both bounds inclusive; we consider $t_{n+1} = +\infty$) when the robot is at point x_i. Count the number of successful commands. Note that it is possible that an ignored command is successful.

Input

The first line contains a single integer t $(1 \le t \le 1000)$ — the number of test cases. The next lines describe the test cases.

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

The next n lines describe the commands. The i-th of these lines contains two integers t_i and x_i $(1 \le t_i \le 10^9, -10^9 \le x_i \le 10^9)$ — the time and the point of the i-th command.

The commands are ordered by time, that is, $t_i < t_{i+1}$ for all possible i.

The sum of n over test cases does not exceed 10^5.

Output

For each testcase output a single integer — the number of successful commands.

input

```
8
3
1 5
3 0
6 4
3
1 5
2 4
10 -5
5
2 -5
3 1
4 1
5 1
6 1
4
3 3
5 -3
9 2
12 0
8
1 1
2 -6
7 2
8 3
12 -9
14 2
18 -1
23 9
5
1 -4
4 -7
6 -1
7 -3
8 -7
2
1 2
2 -2
6
3 10
5 5
8 0
12 -4
14 -7
19 -5
```

output

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

The movements of the robot in the first test case are described in the problem statement. Only the last command is successful.

In the second test case the second command is successful: the robot passes through target point 4 at time 5. Also, the last command is eventually successful.

In the third test case no command is successful, and the robot stops at -5 at time moment 7.

Here are the 0-indexed sequences of the positions of the robot in each second for each testcase of the example. After the cut all the positions are equal to the last one:

1. $[0, 0, 1, 2, 3, 4, 5, 4, 4, \dots]$

2. $[0, 0, 1, 2, 3, 4, 5, 5, 5, 5, 5, 4, 3, 2, 1, 0, -1, -2, -3, -4, -5, -5, \ldots]$
3. $[0, 0, 0, -1, -2, -3, -4, -5, -5, \ldots]$
4. $[0, 0, 0, 0, 1, 2, 3, 3, 3, 3, 2, 2, 2, 1, 0, 0, \ldots]$
5. $[0, 0, 1, 0, -1, -2, -3, -4, -5, -6, -6, -6, -7, -8, -9, -9, -9, -9, -8, -7, -6, -5, -4, -3, -2, -1, -1, \ldots]$
6. $[0, 0, -1, -2, -3, -4, -4, -3, -2, -1, -1, \ldots]$
7. $[0, 0, 1, 2, 2, \ldots]$
8. $[0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, -1, -2, -3, -4, -5, -6, -7, -7, \ldots]$

D. Array Splitting

2 seconds, 256 megabytes

You are given an array $a_1, a_2, \ldots, a_n$ and an integer k.

You are asked to divide this array into k non-empty consecutive subarrays. Every element in the array should be included in exactly one subarray. Let $f(i)$ be the index of subarray the i-th element belongs to. Subarrays are numbered from left to right and from 1 to k.

Let the cost of division be equal to $\sum\limits_{i=1}^{n}(a_i \cdot f(i))$. For example, if $a = [1, -2, -3, 4, -5, 6, -7]$ and we divide it into 3 subbarays in the following way: $[1, -2, -3], [4, -5], [6, -7]$, then the cost of division is equal to $1 \cdot 1 - 2 \cdot 1 - 3 \cdot 1 + 4 \cdot 2 - 5 \cdot 2 + 6 \cdot 3 - 7 \cdot 3 = -9$.

Calculate the maximum cost you can obtain by dividing the array a into k non-empty consecutive subarrays.

Input

The first line contains two integers n and k ($1 \le k \le n \le 3 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($|a_i| \le 10^6$).

Output

Print the maximum cost you can obtain by dividing the array a into k nonempty consecutive subarrays.

input
```
5 2
-1 -2 5 -4 8
```
output
```
15
```

input
```
7 6
-3 0 -1 -2 -2 -4 -1
```
output
```
-45
```

input
```
4 1
3 -1 6 0
```
output
```
8
```

E. Make Palindrome

2 seconds, 256 megabytes

A string is called palindrome if it reads the same from left to right and from right to left. For example "kazak", "oo", "r" and "mikhailrubinchikkihcniburliahkim" are palindroms, but strings "abb" and "ij" are not.

You are given string s consisting of lowercase Latin letters. At once you can choose any position in the string and change letter in that position to any other lowercase letter. So after each changing the length of the string doesn't change. At first you can change some letters in s. Then you can permute the order of letters as you want. Permutation doesn't count as changes.

You should obtain palindrome with the minimal number of changes. If there are several ways to do that you should get the lexicographically (alphabetically) smallest palindrome. So firstly you should minimize the number of changes and then minimize the palindrome lexicographically.

Input

The only line contains string s ($1 \le |s| \le 2 \cdot 10^5$) consisting of only lowercase Latin letters.

Output

Print the lexicographically smallest palindrome that can be obtained with the minimal number of changes.

input
```
aabc
```
output
```
abba
```

input
```
aabcd
```
output
```
abcba
```

F. Education

2 seconds, 256 megabytes

Polycarp is wondering about buying a new computer, which costs c tugriks. To do this, he wants to get a job as a programmer in a big company.

There are n positions in Polycarp's company, numbered starting from one. An employee in position i earns $a[i]$ tugriks every day. The higher the position number, the more tugriks the employee receives. Initially, Polycarp gets a position with the number 1 and has 0 tugriks.

Each day Polycarp can do one of two things:

- If Polycarp is in the position of x, then he can earn $a[x]$ tugriks.
- If Polycarp is in the position of x ($x < n$) and has at least $b[x]$ tugriks, then he can spend $b[x]$ tugriks on an online course and move to the position $x + 1$.

For example, if $n = 4$, $c = 15$, $a = [1, 3, 10, 11]$, $b = [1, 2, 7]$, then Polycarp can act like this:

- On the first day, Polycarp is in the 1-st position and earns 1 tugrik. Now he has 1 tugrik;
- On the second day, Polycarp is in the 1-st position and move to the 2-nd position. Now he has 0 tugriks;
- On the third day, Polycarp is in the 2-nd position and earns 3 tugriks. Now he has 3 tugriks;

- On the fourth day, Polycarp is in the 2-nd position and is transferred to the 3-rd position. Now he has 1 tugriks;
- On the fifth day, Polycarp is in the 3-rd position and earns 10 tugriks. Now he has 11 tugriks;
- On the sixth day, Polycarp is in the 3-rd position and earns 10 tugriks. Now he has 21 tugriks;
- Six days later, Polycarp can buy himself a new computer.

Find the minimum number of days after which Polycarp will be able to buy himself a new computer.

Input

The first line contains a single integer t ($1 \le t \le 10^4$). Then t test cases follow.

The first line of each test case contains two integers n and c ($2 \le n \le 2 \cdot 10^5, 1 \le c \le 10^9$) — the number of positions in the company and the cost of a new computer.

The second line of each test case contains n integers $a_1 \le a_2 \le \ldots \le a_n$ ($1 \le a_i \le 10^9$).

The third line of each test case contains $n-1$ integer $b_1, b_2, \ldots, b_{n-1}$ ($1 \le b_i \le 10^9$).

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the minimum number of days after which Polycarp will be able to buy a new computer.

input
```
3
4 15
1 3 10 11
1 2 7
4 100
1 5 10 50
3 14 12
2 1000000000
1 1
1
```
output
```
6
13
1000000000
```

G. Problem for Nazar

1 second, 256 megabytes

Nazar, a student of the scientific lyceum of the Kingdom of Kremland, is known for his outstanding mathematical abilities. Today a math teacher gave him a `very difficult task`.

Consider two infinite sets of numbers. The first set consists of odd positive numbers ($1, 3, 5, 7, \ldots$), and the second set consists of even positive numbers ($2, 4, 6, 8, \ldots$). At the first stage, the teacher writes the first number on the endless blackboard from the first set, in the second stage — the first two numbers from the second set, on the third stage — the next four numbers from the first set, on the fourth — the next eight numbers from the second set and so on. In other words, at each stage, starting from the second, he writes out **two times more** numbers than at the previous one, and also **changes the set** from which these numbers are written out **to another**.

The ten first written numbers: $1, 2, 4, 3, 5, 7, 9, 6, 8, 10$. Let's number the numbers written, starting **with one**.

The task is to find the sum of numbers with numbers from l to r for given integers l and r. The answer may be big, so you need to find the remainder of the division by 1000000007 ($10^9 + 7$).

Nazar thought about this problem for a long time, but didn't come up with a solution. Help him solve this problem.

Input

The first line contains two integers l and r ($1 \le l \le r \le 10^{18}$) — the range in which you need to find the sum.

Output

Print a single integer — the answer modulo 1000000007 ($10^9 + 7$).

input
```
1 3
```
output
```
7
```

input
```
5 14
```
output
```
105
```

input
```
88005553535 99999999999
```
output
```
761141116
```

In the first example, the answer is the sum of the first three numbers written out ($1 + 2 + 4 = 7$).

In the second example, the numbers with numbers from 5 to 14: $5, 7, 9, 6, 8, 10, 12, 14, 16, 18$. Their sum is 105.

H. Clear the String

3 seconds, 256 megabytes

You are given a string s of length n consisting of lowercase Latin letters. You may apply some operations to this string: in one operation you can delete some contiguous substring of this string, if all letters in the substring you delete are equal. For example, after deleting substring `bbbb` from string `abbbbaccdd` we get the string `aaccdd`.

Calculate the minimum number of operations to delete the whole string s.

Input

The first line contains one integer n ($1 \le n \le 500$) — the length of string s.

The second line contains the string s ($|s| = n$) consisting of lowercase Latin letters.

Output

Output a single integer — the minimal number of operation to delete string s.

input
```
5
abaca
```

output

```
3
```

input

```
8
abcddcba
```

output

```
4
```

I. Rocket

1 second, 256 megabytes

This is an interactive problem.

Natasha is going to fly to Mars. Finally, Natasha sat in the rocket. She flies, flies... but gets bored. She wishes to arrive to Mars already! So she decides to find something to occupy herself. She couldn't think of anything better to do than to calculate the distance to the red planet.

Let's define x as the distance to Mars. Unfortunately, Natasha does not know x. But it is known that $1 \le x \le m$, where Natasha knows the number m. Besides, x and m are positive integers.

Natasha can ask the rocket questions. Every question is an integer y ($1 \le y \le m$). The correct answer to the question is -1, if $x < y$, 0, if $x = y$, and 1, if $x > y$. But the rocket is broken — it does not always answer correctly. Precisely: let the correct answer to the current question be equal to t, then, if the rocket answers this question correctly, then it will answer t, otherwise it will answer $-t$.

In addition, the rocket has a sequence p of length n. Each element of the sequence is either 0 or 1. The rocket processes this sequence in the cyclic order, that is 1-st element, 2-nd, 3-rd, $\ldots$, $(n-1)$-th, n-th, 1-st, 2-nd, 3-rd, $\ldots$, $(n-1)$-th, n-th, $\ldots$. If the current element is 1, the rocket answers correctly, if 0 — lies. Natasha doesn't know the sequence p, but she knows its length — n.

You can ask the rocket no more than 60 questions.

Help Natasha find the distance to Mars. Assume, that the distance to Mars does not change while Natasha is asking questions.

Your solution will not be accepted, if it does not receive an answer 0 from the rocket (even if the distance to Mars is uniquely determined by the already received rocket's answers).

Input

The first line contains two integers m and n ($1 \le m \le 10^9$, $1 \le n \le 30$) — the maximum distance to Mars and the number of elements in the sequence p.

Interaction

You can ask the rocket no more than 60 questions.

To ask a question, print a number y ($1 \le y \le m$) and an end-of-line character, then do the operation `flush` and read the answer to the question.

If the program reads 0, then the distance is correct and you must immediately terminate the program (for example, by calling `exit(0)`). If you ignore this, you can get any verdict, since your program will continue to read from the closed input stream.

If at some point your program reads -2 as an answer, it must immediately end (for example, by calling `exit(0)`). You will receive the "Wrong answer" verdict, and this will mean that the request is incorrect or the number of requests exceeds 60. If you ignore this, you can get any verdict, since your program will continue to read from the closed input stream.

If your program's request is not a valid integer between -2^{31} and $2^{31}-1$ (inclusive) without leading zeros, then you can get any verdict.

You can get "Idleness limit exceeded" if you don't print anything or if you forget to flush the output.

To flush the output buffer you can use (after printing a query and end-of-line):

- `fflush(stdout)` in C++;
- `System.out.flush()` in Java;
- `stdout.flush()` in Python;
- `flush(output)` in Pascal;
- See the documentation for other languages.

Hacking

Use the following format for hacking:

In the first line, print 3 integers m, n, x ($1 \le x \le m \le 10^9$, $1 \le n \le 30$) — the maximum distance to Mars, the number of elements in the sequence p and the current distance to Mars.

In the second line, enter n numbers, each of which is equal to 0 or 1 — sequence p.

The hacked solution will not have access to the number x and sequence p.

input

```
5 2
1
-1
-1
1
0
```

output

```
1
2
4
5
3
```

In the example, hacking would look like this:

```
5 2 3

1 0
```

This means that the current distance to Mars is equal to 3, Natasha knows that it does not exceed 5, and the rocket answers in order: correctly, incorrectly, correctly, incorrectly ...

Really:

on the first query (1) the correct answer is 1, the rocket answered correctly: 1;

on the second query (2) the correct answer is 1, the rocket answered incorrectly: -1;

on the third query (4) the correct answer is -1, the rocket answered correctly: -1;

on the fourth query (5) the correct answer is -1, the rocket answered incorrectly: 1;

on the fifth query (3) the correct and incorrect answer is 0.

J. Zuma

2 seconds, 512 megabytes

Genos recently installed the game Zuma on his phone. In Zuma there exists a line of n gemstones, the i-th of which has color c_i. The goal of the game is to destroy all the gemstones in the line as quickly as possible.

In one second, Genos is able to choose exactly one continuous substring of colored gemstones that is a palindrome and remove it from the line. After the substring is removed, the remaining gemstones shift to form a solid line again. What is the minimum number of seconds needed to destroy the entire line?

Let us remind, that the string (or substring) is called *palindrome*, if it reads same backwards or forward. In our case this means the color of the first gemstone is equal to the color of the last one, the color of the second gemstone is equal to the color of the next to last and so on.

Input

The first line of input contains a single integer n $(1 \leq n \leq 500)$ — the number of gemstones.

The second line contains n space-separated integers, the i-th of which is c_i $(1 \leq c_i \leq n)$ — the color of the i-th gemstone in a line.

Output

Print a single integer — the minimum number of seconds needed to destroy the entire line.

input

```
3
1 2 1
```

output

```
1
```

input

```
3
1 2 3
```

output

```
3
```

input

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

output

```
2
```

In the first sample, Genos can destroy the entire line in one second.

In the second sample, Genos can only destroy one gemstone at a time, so destroying three gemstones takes three seconds.

In the third sample, to achieve the optimal time of two seconds, destroy palindrome `4 4` first and then destroy palindrome `1 2 3 2 1`.