# Educational Codeforces Round 109 (Rated for Div. 2)

## A. Potion-making

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have an initially empty cauldron, and you want to brew a potion in it. The potion consists of two ingredients: magic essence and water. The potion you want to brew should contain exactly $\%$ magic essence and $(100 - )\ \%$ water.

In one step, you can pour either one liter of magic essence or one liter of water into the cauldron. What is the minimum number of steps to brew a potion? You don't care about the total volume of the potion, only about the ratio between magic essence and water in it.

A small reminder: if you pour liters of essence and liters of water ( $+ > 0$) into the cauldron, then it contains $\frac{}{+} \cdot 100\ \%$ (without rounding) magic essence and $\frac{}{+} \cdot 100\ \%$ water.

### Input

The first line contains the single $(1 \le \le 100)$ — the number of test cases.

The first and only line of each test case contains a single integer $(1 \le \le 100)$ — the percentage of essence in a good potion.

### Output

For each test case, print the minimum number of steps to brew a good potion. It can be proved that it's always possible to achieve it in a finite number of steps.

**Example**

| input |
| --- |
| 3<br>3<br>100<br>25 |
| **output** |
| 100<br>1<br>4 |

### Note

In the first test case, you should pour $3$ liters of magic essence and $97$ liters of water into the cauldron to get a potion with $3\ \%$ of magic essence.

In the second test case, you can pour only $1$ liter of essence to get a potion with $100\ \%$ of magic essence.

In the third test case, you can pour $1$ liter of magic essence and $3$ liters of water.

## B. Permutation Sort

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a permutation consisting of numbers $1$, $2$, ..., (a permutation is an array in which each element from $1$ to occurs exactly once).

You can perform the following operation: choose some subarray (contiguous subsegment) of and rearrange the elements in it in any way you want. But this operation cannot be applied to the whole array.

For example, if $= [2, 1, 4, 5, 3]$ and we want to apply the operation to the subarray $[2, 4]$ (the subarray containing all elements from the $2$-nd to the $4$-th), then after the operation, the array can become $= [2, 5, 1, 4, 3]$ or, for example, $= [2, 1, 5, 4, 3]$.

Your task is to calculate the minimum number of operations described above to sort the permutation in ascending order.

### Input

The first line contains a single integer $(1 \le \le 2000)$ — the number of test cases.

The first line of the test case contains a single integer $(3 \le \le 50)$ — the number of elements in the permutation.

The second line of the test case contains distinct integers from $1$ to — the given permutation .

### Output
For each test case, output a single integer — the minimum number of operations described above to sort the array in ascending order.

**Example**

| input |
| --- |
| 3<br>4<br>1 3 2 4<br>3<br>1 2 3<br>5<br>2 1 4 5 3 |
| **output** |
| 1<br>0<br>2 |

### Note
In the explanations, $[\ ,\ ]$ defines the subarray of that starts from the -th element and ends with the -th element.

In the first test case of the example, you can select the subarray $[2, 3]$ and swap the elements in it.

In the second test case of the example, the permutation is already sorted, so you don't need to apply any operations.

In the third test case of the example, you can select the subarray $[3, 5]$ and reorder the elements in it so becomes $[2, 1, 3, 4, 5]$, and then select the subarray $[1, 2]$ and swap the elements in it, so becomes $[1, 2, 3, 4, 5]$.

## C. Robot Collisions

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are robots driving along an OX axis. There are also two walls: one is at coordinate $0$ and one is at coordinate .

The -th robot starts at an integer coordinate $(0 <$ $<$ $)$ and moves either left (towards the $0$) or right with the speed of $1$ unit per second. No two robots start at the same coordinate.

Whenever a robot reaches a wall, it turns around instantly and continues his ride in the opposite direction with the same speed.

Whenever several robots meet at the same **integer** coordinate, they collide and explode into dust. Once a robot has exploded, it doesn't collide with any other robot. Note that if several robots meet at a non-integer coordinate, nothing happens.

For each robot find out if it ever explodes and print the time of explosion if it happens and $-1$ otherwise.

### Input
The first line contains a single integer $(1 \le$ $\le 1000)$ — the number of testcases.

Then the descriptions of testcases follow.

The first line of each testcase contains two integers and $(1 \le$ $\le 3 \cdot 10^5; 2 \le$ $\le 10^8)$ — the number of robots and the coordinate of the right wall.

The second line of each testcase contains integers $_1, \ _2, \ldots,$ $(0 <$ $<$ $)$ — the starting coordinates of the robots.

The third line of each testcase contains space-separated characters 'L' or 'R' — the starting directions of the robots ('L' stands for left and 'R' stands for right).

All coordinates in the testcase are distinct.

The sum of over all testcases doesn't exceed $3 \cdot 10^5$.

### Output
For each testcase print integers — for the -th robot output the time it explodes at if it does and $-1$ otherwise.

**Example**

| input |
| --- |
| 5<br>7 12<br>1 2 3 4 9 10 11<br>R R L L R R R<br>2 10<br>1 6<br>R R<br>2 10<br>1 3<br>L L |

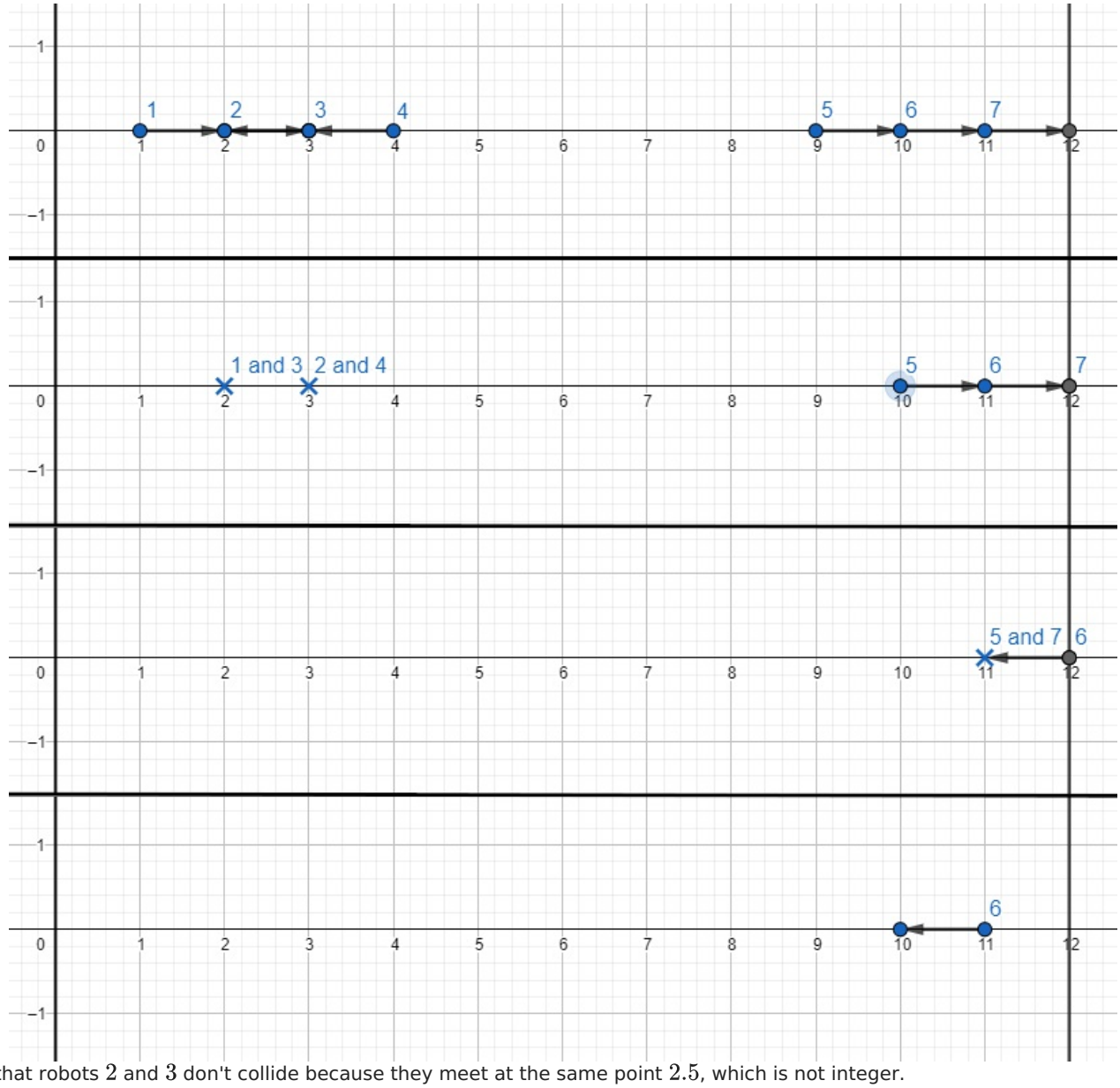
```
1 10
5
R
7 8
6 1 7 2 3 5 4
R L R L L L L
```

**output**

```
1 1 1 1 2 -1 2
-1 -1
2 2
-1
-1 2 7 3 2 7 3
```

### Note

Here is the picture for the seconds $0, 1, 2$ and $3$ of the first testcase:




Notice that robots $2$ and $3$ don't collide because they meet at the same point $2.5$, which is not integer.

After second $3$ robot $6$ just drive infinitely because there's no robot to collide with.

## D. Armchairs

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

There are armchairs, numbered from $1$ to from left to right. Some armchairs are occupied by people (at most one person per armchair), others are not. The number of occupied armchairs is not greater than $\frac{}{2}$.

For some reason, you would like to tell people to move from their armchairs to some other ones. If the -th armchair is occupied by someone and the -th armchair is not, you can tell the person sitting in the -th armchair to move to the -th armchair. The time it takes a person to move from the -th armchair to the -th one is $|\ -\ |$ minutes. You may perform this operation any number of times, but these operations must be done sequentially, i. e. you cannot tell a person to move until the person you asked to move in the last operation has finished moving to their destination armchair.

You want to achieve the following situation: every seat that was initially occupied must be free. What is the minimum time you need to do it?

### Input
The first line contains one integer $(2 \leq \leq 5000)$ — the number of armchairs.

The second line contains integers $_1, _2, \ldots,$ $(0 \leq \leq 1)$. $= 1$ means that the -th armchair is initially occupied, $= 0$ means that it is initially free. The number of occupied armchairs is at most $\frac{}{2}$.

### Output
Print one integer — the minimum number of minutes you have to spend to achieve the following situation: every seat that was initially occupied must be free.

**Examples**

| input |
| --- |
| 7<br>1 0 0 1 0 0 1 |
| **output** |
| 3 |

| input |
| --- |
| 6<br>1 1 1 0 0 0 |
| **output** |
| 9 |

| input |
| --- |
| 5<br>0 0 0 0 0 |
| **output** |
| 0 |

### Note
In the first test, you can perform the following sequence:

1. ask a person to move from armchair $1$ to armchair $2$, it takes $1$ minute;
2. ask a person to move from armchair $7$ to armchair $6$, it takes $1$ minute;
3. ask a person to move from armchair $4$ to armchair $5$, it takes $1$ minute.

In the second test, you can perform the following sequence:

1. ask a person to move from armchair $1$ to armchair $4$, it takes $3$ minutes;
2. ask a person to move from armchair $2$ to armchair $6$, it takes $4$ minutes;
3. ask a person to move from armchair $4$ to armchair $5$, it takes $1$ minute;
4. ask a person to move from armchair $3$ to armchair $4$, it takes $1$ minute.

In the third test, no seat is occupied so your goal is achieved instantly.

## E. Assimilation IV

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Monocarp is playing a game "Assimilation IV". In this game he manages a great empire: builds cities and conquers new lands.

Monocarp's empire has cities. In order to conquer new lands he plans to build **one Monument in each city**. The game is turn-based and, since Monocarp is still amateur, he builds exactly one Monument per turn.

Monocarp has points on the map he'd like to control using the constructed Monuments. For each point he knows the distance between it and each city. Monuments work in the following way: when built in some city, a Monument controls all points at distance at most $1$ to this city. Next turn, the Monument controls all points at distance at most $2$, the turn after — at distance at most $3$, and so on. Monocarp will build Monuments in turns and his empire will conquer all points that are controlled by at least one Monument.

Monocarp can't figure out any strategy, so during each turn he will choose a city for a Monument randomly among all remaining cities (cities without Monuments). Monocarp wants to know how many points (among of them) he will conquer at the end of turn number . Help him to calculate the expected number of conquered points!

### Input
The first line contains two integers and $(1 \leq \leq 20; 1 \leq \leq 5 \cdot 10^4)$ — the number of cities and the number of points.

Next  lines contains  integers each: the  -th integer of the  -th line  , ($1 \leq$  , $\leq$  $+1$) is the distance between the  -th city and the  -th point.

### Output

It can be shown that the expected number of points Monocarp conquers at the end of the  -th turn can be represented as an irreducible fraction —. Print this fraction modulo $998\,244\,353$, i. e. value  $\cdot$  $^{-1} \bmod 998244353$ where  $^{-1}$ is such number that  $\cdot$  $^{-1} \bmod 998244353 = 1$.

#### Example

| input |
| --- |
| 3 5<br>1 4 4 3 4<br>1 4 1 4 2<br>1 4 4 4 3 |
| **output** |
| 166374062 |

### Note

Let's look at all possible orders of cities Monuments will be build in:

- $[1, 2, 3]$:
  - the first city controls all points at distance at most $3$, in other words, points $1$ and $4$;
  - the second city controls all points at distance at most $2$, or points $1$, $3$ and $5$;
  - the third city controls all points at distance at most $1$, or point $1$.

  In total, $4$ points are controlled.
- $[1, 3, 2]$: the first city controls points $1$ and $4$; the second city — points $1$ and $3$; the third city — point $1$. In total, $3$ points.
- $[2, 1, 3]$: the first city controls point $1$; the second city — points $1$, $3$ and $5$; the third city — point $1$. In total, $3$ points.
- $[2, 3, 1]$: the first city controls point $1$; the second city — points $1$, $3$ and $5$; the third city — point $1$. In total, $3$ points.
- $[3, 1, 2]$: the first city controls point $1$; the second city — points $1$ and $3$; the third city — points $1$ and $5$. In total, $3$ points.
- $[3, 2, 1]$: the first city controls point $1$; the second city — points $1$, $3$ and $5$; the third city — points $1$ and $5$. In total, $3$ points.

The expected number of controlled points is $\frac{4+3+3+3+3+3}{6} = \frac{19}{6}$ or $19 \cdot 6^{-1} \equiv 19 \cdot 166374059 \equiv 166374062 \pmod{998244353}$

## F. Goblins And Gnomes

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Monocarp plays a computer game called "Goblins and Gnomes". In this game, he manages a large underground city of gnomes and defends it from hordes of goblins.

The city consists of  halls and  one-directional tunnels connecting them. The structure of tunnels has the following property: if a goblin leaves any hall, he cannot return to that hall.

The city will be attacked by  waves of goblins; during the  -th wave,  goblins attack the city. Monocarp's goal is to pass all waves.

The  -th wave goes as follows: firstly,  goblins appear in some halls of the city and pillage them; **at most one goblin appears in each hall**. Then, goblins start moving along the tunnels, pillaging all the halls in their path.

Goblins are very greedy and cunning, so they choose their paths so that no two goblins pass through the same hall. Among all possible attack plans, they choose a plan which allows them to **pillage the maximum number of halls**. After goblins are done pillaging, they leave the city.

If all halls are pillaged during the wave — Monocarp loses the game. Otherwise, the city is restored. If some hall is pillaged during a wave, goblins are still interested in pillaging it during the next waves.

Before each wave, Monocarp can spend some time preparing to it. Monocarp doesn't have any strict time limits on his preparations (he decides when to call each wave by himself), but the longer he prepares for a wave, the fewer points he gets for passing it. If Monocarp prepares for the  -th wave for  minutes, then he gets $\max(0,$  $-$  $\cdot$  $)$ points for passing it (obviously, if he doesn't lose in the process).

While preparing for a wave, Monocarp can block tunnels. He can spend one minute to **either block all tunnels leading from some hall or block all tunnels leading to some hall**. If Monocarp blocks a tunnel while preparing for a wave, it stays blocked during the next waves as well.

Help Monocarp to defend against all  waves of goblins and get the maximum possible amount of points!

### Input

The first line contains three integers  ,  and  ($2 \leq$  $\leq 50$; $0 \leq$  $\leq \frac{(\ -1)}{2}$; $1 \leq$  $\leq$  $-1$) — the number of halls in the city,

the number of tunnels and the number of goblin waves, correspondely.

Next lines describe tunnels. The -th line contains two integers and ($1 \leq$ , $\leq$ ; $\neq$ ). It means that the tunnel goes from hall to hall . **The structure of tunnels has the following property: if a goblin leaves any hall, he cannot return to that hall**. There is at most one tunnel between each pair of halls.

Next lines describe the scoring system. The -th line contains two integers and ($1 \leq$ $\leq 10^9$; $1 \leq$ $\leq 10^9$). If Monocarp prepares for the -th wave for minutes, then he gets $\max(0,$ $-$ $\cdot$ $)$ points for passing it.

## Output

Print the optimal Monocarp's strategy in the following format:

At first, print one integer ( $\leq$ $\leq 2$ $+$ ) — the number of actions Monocarp will perform. Next, print actions themselves in the order Monocarp performs them. The -th action is described by a single integer ($-$ $\leq$ $\leq$ ) using the following format:

- if $> 0$ then Monocarp blocks all tunnels going out from the hall ;
- if $< 0$ then Monocarp blocks all tunnels going into the hall $|$ $|$;
- if $= 0$ then Monocarp calls the next goblin wave.

You can't repeat the same block action several times. Monocarp must survive all waves he calls (goblins shouldn't be able to pillage all halls). Monocarp should call exactly waves and earn the maximum possible number of points in total.

If there are several optimal strategies — print any of them.

**Examples**

| input |
| --- |
| 5 4 4<br>1 2<br>2 3<br>4 3<br>5 3<br>100 1<br>200 5<br>10 10<br>100 1 |
| **output** |
| 6<br>-2 -3 0 0 0 0 |

| input |
| --- |
| 5 4 4<br>1 2<br>2 3<br>4 3<br>5 3<br>100 100<br>200 5<br>10 10<br>100 1 |
| **output** |
| 6<br>0 -3 0 0 1 0 |

| input |
| --- |
| 5 10 1<br>1 2<br>1 3<br>1 4<br>1 5<br>5 2<br>5 3<br>5 4<br>4 2<br>4 3<br>2 3<br>100 100 |
| **output** |
| 6<br>1 2 3 4 5 0 |

## Note

In the first example, Monocarp, firstly, block all tunnels going in hall $2$, secondly — all tunnels going in hall $3$, and after that calls all waves. He spent two minutes to prepare to wave $1$, so he gets $98$ points for it. He didn't prepare after that, that's why he gets maximum scores for each of next waves ($200$, $10$ and $100$). In total, Monocarp earns $408$ points.

In the second example, Monocarp calls for the first wave immediately and gets $100$ points. Before the second wave he blocks all tunnels going in hall $3$. He spent one minute preparing to the wave, so he gets $195$ points. Monocarp didn't prepare for the third wave, so he gets $10$ points by surviving it. Before the fourth wave he blocks all tunnels going out from hall $1$. He spent one minute,

so he gets $99$ points for the fourth wave. In total, Monocarp earns $404$ points.

In the third example, it doesn't matter how many minutes Monocarp will spend before the wave, since he won't get any points for it. That's why he decides to block all tunnels in the city, spending $5$ minutes. He survived the wave though without getting any points.

---