regionals | latin america region

icpc international collegiate programming contest

icpc.foundation

icpc global sponsor programming tools

HUAWEI icpc diamond multi-regional sponsor

International Collegiate Programming Contest 2023

Latin American Regional Contests

October 21, 2023

Warmup Session

This problem set contains 3 problems; pages are numbered from 1 to 6.

This problem set is used in simultaneous contests with the following participating countries:

Antigua y Barbuda, Argentina, Bolivia, Brasil, Chile, Colombia, Costa Rica, Cuba, El Salvador, Guatemala, México, Perú, Puerto Rico, República Dominicana, Trinidad y Tobago, Uruguay and Venezuela

v1.0

General information

Unless otherwise stated, the following conditions hold for all problems.

Program name

1. Your solution must be called *codename*`.c`, *codename*`.cpp`, *codename*`.java`, *codename*`.kt`, *codename*`.py3`, where *codename* is the capital letter which identifies the problem.

Input

1. The input must be read from standard input.
2. The input consists of a single test case, which is described using a number of lines that depends on the problem. No extra data appear in the input.
3. When a line of data contains several values, they are separated by *single* spaces. No other spaces appear in the input. There are no empty lines.
4. The English alphabet is used. There are no letters with tildes, accents, diaereses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, has the usual end-of-line mark.

Output

1. The output must be written to standard output.
2. The result of the test case must appear in the output using a number of lines that depends on the problem. No extra data should appear in the output.
3. When a line of results contains several values, they must be separated by *single* spaces. No other spaces should appear in the output. There should be no empty lines.
4. The English alphabet must be used. There should be no letters with tildes, accents, diaereses or other diacritical marks (ñ, Ã, é, Ì, ô, Ü, ç, etcetera).
5. Every line, including the last one, must have the usual end-of-line mark.
6. To output real numbers, round them to the closest rational with the required number of digits after the decimal point. Test case is such that there are no ties when rounding as specified.

Problem A – Invested Money

Nowadays your programming skills are amazing, and you regularly receive lots of money for your work. Unfortunately, your financial skills did not evolve the same way. So each time you earn some money, you simply invest it in a bank in a 30 days time deposit with an automatic renewal clause. This means that 30 days after you invest the money, it is invested for 30 additional days, over and over again, until you inform the bank that you want to stop the renewal and get your money back. Time deposits cannot be created nor renewed during weekends; if a 30 days period ends on a weekend, the renewal occurs on the immediately following Monday.

Since the bank holds almost all your money, you must wait until the closest renewal each time you want to buy anything but daily food. Today you decided to buy a new smartphone to replace your six-month-old device. Given the dates when you created each time deposit, you must determine the minimum number of days that you must wait to get some money from the bank.

As an example, suppose that today is Saturday and that you created five time deposits: a time deposit last Monday, another time deposit last Tuesday, yet another time deposit last Wednesday, and two time deposits yesterday. The first time deposit (Monday) would be renewed on a Wednesday after 25 days from today. The second time deposit (Tuesday) would be renewed on a Thursday after 26 days from today. The third time deposit (Wednesday) would be renewed on a Friday after 27 days from today. Finally, the last two time deposits (Friday) would be renewed on a Monday after 30 days from today, because the renewal on a Sunday is not allowed. Thus, in this case, you must wait 25 days to get some money from the bank.

Input

The first line contains a string T and an integer N ($1 \leq N \leq 10^5$), indicating respectively today's day of the week and the number of time deposits. The string is either "`Mon`", "`Tue`", "`Wed`", "`Thu`", "`Fri`", "`Sat`", or "`Sun`", representing respectively that today is Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, or Sunday. The second line contains N integers $D_1, D_2, \ldots, D_N$ ($0 \leq D_i \leq 10^9$ for $i = 1, 2, \ldots, N$), indicating the number of days elapsed since each time deposit was created. It is guaranteed that the time deposits were not created during weekends.

Output

Output a single line with an integer indicating the minimum number of days that you must wait to get some money from the bank.

Sample input 1	**Sample output 1**

`Sat 5`
`5 4 3 1 1`
	`25`

Sample input 2	**Sample output 2**

`Sat 5`
`3 1 4 1 5`
	`25`

Sample input 3	**Sample output 3**

`Thu 1`
`0`
	`32`

Sample input 4	**Sample output 4**

`Thu 1`
`30`
	`0`

Sample input 5	**Sample output 5**

`Fri 1`
`31`
	`31`

This page would be intentionally left blank if we would not wish to inform about that.

Problem B – Daily Trips

Bella is a simple girl with a simple life: wake up, go to work, work, go home, rest, sleep, and repeat. Bella commutes via bus, and it rains often in her city, so sometimes she needs an umbrella. However, the local weather forecast is unreliable, so Bella can't be sure if it's going to rain or not until right before she begins a trip. To avoid getting caught unprepared, Bella created a system.

She owns one or more umbrellas, keeping them at home or her workplace. Before any trip (from home to work, or vice versa), Bella looks outside to decide whether to bring an umbrella on that trip:

- if it's raining, she brings an umbrella;
- otherwise, if there are currently no umbrellas at her destination (either work or home), she still brings an umbrella, just in case;
- otherwise, she doesn't bring an umbrella.

The second rule above is meant to prevent a situation where Bella needs an umbrella but has none at her current location (a bad memory she will talk about to anyone who asks). This guarantees that Bella will never catch rain and get sick.

Now we need you to simulate Bella's method for a certain period. The simulation starts with Bella at home. Each day she takes two bus trips: to and back from work. Given the starting numbers of umbrellas at her home and workplace, and the weather reports during N consecutive days, find out whether or not Bella brought an umbrella on each of her $2N$ bus trips.

Input

The first line contains three integers N ($1 \leq N \leq 10^4$), H ($1 \leq H \leq 100$), and W ($0 \leq W \leq 100$), indicating respectively the duration of the simulation period in days, and the starting numbers of umbrellas at Bella's home and workplace. For $i = 1, 2, \ldots, N$, the i-th of the next N lines contains two characters representing whether it rained on each trip of the i-th day. The first character refers to the first trip of the day (from home to work), while the second character refers to the second trip of the day (from work to home). Each character is the uppercase letter "`Y`" if it rained, and the uppercase letter "`N`" otherwise.

Output

Output N lines. For $i = 1, 2, \ldots, N$, the i-th line must contain two characters indicating whether Bella brought an umbrella on each trip of the i-th day. The first character refers to the first trip while the second character refers to the second trip. Each character must be the uppercase letter "`Y`" if Bella brought an umbrella, and the uppercase letter "`N`" otherwise.

Sample input 1	**Sample output 1**

```
5 2 1
Y N
N N
Y N
N Y
Y Y
```
	```
	Y N
	N N
	Y Y
	N Y
	Y Y
	```

This page would be intentionally left blank if we would not wish to inform about that.

Problem C – Italian Calzone & Pasta Corner

The Italian Calzone & Pasta Corner restaurant designed its menu having its dishes in a $R \times C$ two-dimensional grid, keeping dishes that go well together nearby each other. To eat, you choose a starting cell, and then repeatedly move up, down, left, or right to any of the four adjacent cells, getting any dishes you move through. Moving into already visited cells is allowed, but you don't get the same dish again.

One day, Pierre, a foreign customer, showed up really hungry, and with a very strict etiquette background. He has a very specific order in which the dishes must be eaten. For example an appetizer, then an entree, then a main dish, then a salad, etc. So he assigned a distinct integer from 1 to $R \times C$ to each dish in the menu grid, indicating the order in which he would eat the whole menu. Now he wants to choose and eat dishes following his order. Since the restaurant's rules might prevent him from choosing the whole menu, he is fine with skipping some of the steps given by his order. Can you help him choose a meal with as many dishes as possible?

Input

The first line contains two integers R and C ($1 \leq R, C \leq 100$), indicating that the menu grid has R rows and C columns. The next R lines contain C integers each, representing the menu grid. Each of these numbers is a distinct integer from 1 to $R \times C$ assigned by Pierre to the corresponding dish in the menu grid.

Output

Output a single line with an integer indicating the maximum amount of dishes that Pierre can eat.

Sample input 1	**Sample output 1**

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

Sample input 2	**Sample output 2**

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

Sample input 3	**Sample output 3**

`3 3`
`4 1 3`
`8 5 9`
`7 2 6`
	`6`

This page would be intentionally left blank if we would not wish to inform about that.