ECOO 2017

Programming Contest Questions

Regional Competition (Round 2)

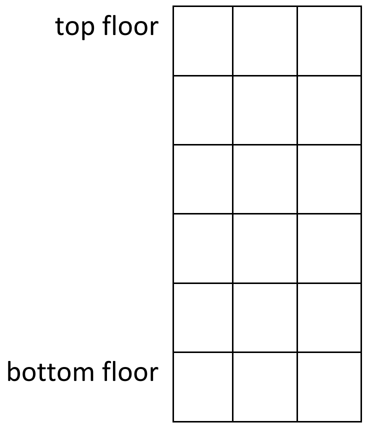
April 29, 2017

Problem 1: Wizard’s Windows

Author: Andrew Seidel

Whizzy the Wizard is trying to decide on which new mage tower to move into. Although each tower is made up of different shapes and sizes, the towers are all completely made of glass. Some of the glass panels are windows *(W)* that can be opened magically, but some of the glass panels are solid panes of glass *(G)* in order to provide structural support.

Whizzy has access to every glass panel and he is trying to figure out which panels are windows. Whizzy has gone through two floors of windows and is getting tired of trying to figure out which is which. He is hoping to open the top floor’s windows, so he needs to find a pattern to help him out.

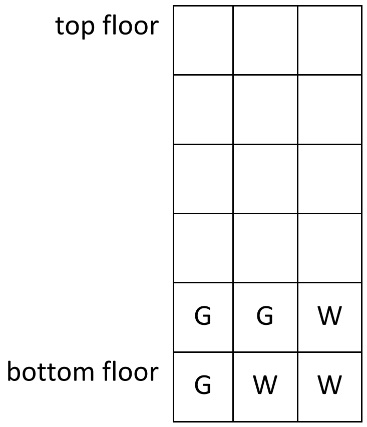


If we lay out a net of each of the towers, they form a net that is *N* by *M* in dimension where the width of the building is defined by *N* and the number of floors in the building is represented by *M*. A sample diagram as left *(assuming N = 3, M = 6)*:

Whizzy finds that the text written on the front door of the building is a key to figuring out the pattern. An example key is shown on the right.

Key:
```
GW ► G
GG ► W
WG ► W
WW ► G
```

To read the key, the letter G represents a support piece of glass, the letter W represents a window. The letter pairs describe what to do to the cell above, based on the two letters surrounding the current position.



For example, take a look at the bottom floor of the tower to the right. The middle square has a G on the left, and a W on the right. So, according to the first rule in the key (“`GW ► G`”), the square above it has to be a G. The left-most square is surrounded by a W on the left, (because the tower net wraps around), and a W on the right. So the last rule in the key makes the square above it a G. The right-most square has a W to the left of it, and a G to the right of it, making the square above a W.

Input Specifications

DATA11.txt (DATA12.txt for the second try) will contain 10 towers, each terminated with a line containing a single asterisk. For each tower, the input will be formatted as follows:

- The first line contains **N**, (3 ≤ **N** ≤ 8) and **M**, (4 ≤ **M** ≤ 1000) separated by a space.
- The next 4 lines will contain the rules for moving up the tower, with the left and right side of each rule separated by a space.
- The next line contains the layout of the bottom floor.

Output Specifications

Output the value of the top floor of the tower as a single, contiguous set of capital letters.

Sample Input (2 towers shown)

```
3 6
WW G
WG G
GW W
GG W
GWG
*
3 4
WW W
WG W
GW G
GG G
GGW
*
```

Sample Output

```
GWW
GGW
```

ECOO 2017 Question Development Team

Kevin Forest	Sheridan College
John Ketelaars	ECOO-CS Communications
Stella Lau	University of Cambridge
Greg Reid	St. Francis Xavier Secondary School, Mississauga
Sam Scott	Mohawk College
Andrew Seidel	John Fraser Secondary School, Mississauga
David Stermole	ECOO-CS President
Reyno Tilikaynen	University of Waterloo

Problem 2: Treasure Hunt

Author: Reyno Tilikaynen

Sanjay and his friends are searching for treasure in an abandoned castle. They have just entered the castle and find themselves faced with a maze of treasure-filled rooms separated by locked doors. Fortunately for them, the old owners left spare keys littered around the castle.

Sanjay finds that they can use any key to help open any door in the castle. However, some doors require multiple keys in order to open them.

The floor plan of the castle can be represented as an **N** x **N** grid of the following characters:

- '`.`' denotes an empty space
- '#' denotes a wall.
- Characters '1' through '9' denote a doors, with the number representing how many keys it takes to open the door.
- 'K' represents a key.
- 'T' represents treasure.
- 'S' represents Sanjay's start point.

Input Specifications

DATA21.txt (DATA22.txt for the second try) will contain 10 test cases. Each test case starts with an integer **N** (1 ≤ **N** ≤ 1000). The next **N** lines each contain **N** characters denoting the floor plan of the castle.

Output Specifications

For each test case, your program should output the number of treasures that Sanjay and his friends will be able to retrieve.

Sample Input (2 cases shown):

```
3
.SK
#1#
T..
5
S.2.T
.K##3
11##T
....#
..T.K
```

Sample Output:

```
1
2
```

Problem 3: Lunch Time

Author: Reyno Tilikaynen

Since beginning her university studies, Ava has been going to a nearby plaza for lunch. In order to maintain good eating habits throughout her university career, Ava has crafted a perfect dining strategy.

Ava began by eating at each of the **N** restaurants (numbered 1 through **N**) at the plaza and rating how much she enjoyed the food. Once she gathered all the ratings, she would only eat at the highest-rated restaurant. (If there is a tie, she eats at the lowest-numbered restaurant.) However, after a week of eating the same food, Ava realized she needs more variety in her diet. To fix this issue, she decided that eating at a restaurant would cause its rating to drop by a fixed amount, **M**.

Armed with her dining strategy, Ava wonders where she will grab lunch on her last day of university, which is **K** days away if she eats at exactly one restaurant per day.

Input Specifications

DATA31.txt (DATA32.txt for the second try) will contain 10 test cases. Each test case starts with three integers **N, M**, **K** ($1 \le \mathbf{N}, \mathbf{M} \le 100{,}000$, $1 \le \mathbf{K} \le 10^{12}$). The next line contains **N** positive integers $\mathbf{R_p}$ ($1 \le \mathbf{R_p} \le 10^9$), where $\mathbf{R_p}$ represents the rating of the $\mathbf{p^{th}}$ restaurant at the plaza. Restaurants are numbered starting from 1.

For the first four test cases in the file, $\mathbf{N*K} \le 10^6$. For the first seven cases, $\mathbf{K} \le 10^6$.

Output Specifications

For each test case, your program should output one integer representing the restaurant Ava will eat at on the $\mathbf{K}^{th}$ day.

Sample Input (3 cases shown):

```
2 5 4
20 17
5 4 7
1 2 4 8 16
4 8 100
3 22 20 14
```

Sample Output:

```
2
5
3
```

Problem 4: Zig Zag

Author: Reyno Tilikaynen

A sequence of integers from 1 to N is said to be zig zag if the sequence contains each integer exactly once and, indexing from one, every element at an even index is greater than the preceding one, and every element at an odd index is less than the preceding one.

For example:

1, 3, 2, 5, 4	is zig zag.
2, 5, 3, 4, 1	is zig zag
1, 5, 4, 2, 3	is NOT zig zag (the fourth element is less than the third element).

Input Specifications

DATA41.txt (DATA42.txt for the second try) will contain 10 test cases. Each test case has one line with the integer **N** (1 ≤ **N** ≤ 10,000). For the first four test cases in the file, **N** ≤ 10. For the first seven cases, **N** ≤ 200.

Output Specifications

For each test case, output the number of unique zig zag sequences possible for that value of **N**, modulo 1,000,000,007. “Modulo 1,000,000,007” means that if the answer is 123,456,789,123 you should output 456,788,262 because that’s the remainder of 123,456,789,123 divided by 1,000,000,007.

Sample Input (3 cases shown)

```
3
4
5
```

Sample Output

```
2
5
16
```