Problem 215: Turing Interpreter

Difficulty: Medium

Author: Louis Ronat, Denver, Colorado, United States

Originally Published: Code Quest 2023

Problem Background

Alan Turing is widely credited with creating the first mathematical model of a computer. His model, the Turing Machine, can be thought of as a mechanical machine which holds a strip of paper called a tape. Based on the machine's internal state and the data printed on the tape at the current position, the machine will write new data to the tape, move the tape forward or backwards one step, and update its internal state. The Turing Machine forms the basis of defining whether a machine is "Turing Complete," which is an important determination to be made about programming languages.

Problem Description

Turing Machines take in a set of instructions which describe what to do in each situation the machine might encounter. Each instruction includes five values:

1. $q_{current}$ - The current machine state in which the instruction applies
2. s_{read} - The tape symbol the machine is currently reading
3. s_{write} - The symbol that should be printed on the tape in that position (overwriting s_{read})
4. d - The direction in which to move the tape reader one position after printing (L for left, R for right)
5. q_{new} - The new machine state to set

These values can be combined into a compact form called a 5-tuple:

$$\{q_{current}, s_{read}, s_{write}, d, q_{new}\}$$

A set of 5-tuples forms the complete set of instructions for a Turing Machine, and is known as a Turing Table. A sample Turing Table and its interpretation is shown below:

5-tuple	If the machine is in state...	...and reads the symbol...	...print this symbol in its place...	...then move one space to the...	...and set this as the new machine state
{A, 0, 1, R, A}	A	0	1	right	A
{A, 1, 1, L, B}	A	1	1	left	B
{B, 1, 0, L, B}	B	1	0	left	B
{B, 0, 0, R, A}	B	0	0	right	A

Let’s demonstrate how this machine works in practice. The tables below represent a tape being read by the Turing Machine described above. The black cell represents the position of the reader. The machine starts in state A.

0	1	1	**0**	0	0	1

The machine reads in a 0, and so executes the first instruction listed above. It writes a 1, then moves one position to the right, and remains in state A. After this first step, the tape now looks like this:

0	1	1	1	**0**	0	1

The machine is still in state A and is reading another 0. This causes the machine to execute the same instruction again. After the completion of that step, it again finds itself in the same situation, and repeats the first instruction one more time.

0	1	1	1	1	**0**	1
0	1	1	1	1	1	**1**

Now the situation has changed. The machine is still in state A, but now it’s reading a 1. It now executes the second instruction: it writes a 1, then moves one position to the left, then sets the state to B.

0	1	1	1	1	**1**	1

With the machine now in state B, it reads a 1, and follows the third instruction: write a 0, move to the left, and remain in state B. This continues until the machine encounters a 0.

0	1	1	1	**1**	0	1
0	1	1	**1**	0	0	1
0	1	**1**	0	0	0	1
0	**1**	0	0	0	0	1
0	0	0	0	0	0	1

With the zero now reached, the machine executes its final instruction: write a 0, move to the right, and set the state to A.

0	**0**	0	0	0	0	1

We now find ourselves in the same situation we started in: in state A, reading a 0. The machine now enters an infinite loop, writing 1’s to the tape as it moves to the right, then writing 0’s as it moves to the left.

For this problem, you will need to implement your own Turing Machine that follows the instructions provided in a given Turing Table over a certain number of steps. You may operate under the following guarantees:

- The machines you implement will never move beyond the edges of the provided tape.
- Machines will always start in state A.
- Machines will always start in the exact middle of the tape, which will always contain an odd number of symbols.
- The provided 5-tuples will cover every possible scenario the machine may encounter.

Sample Input

The first line of your program's input, **received from the standard input channel**, will contain a positive integer representing the number of test cases. Each test case will include:

- A line containing a single positive integer, T, representing the number of instructions contained in the Turing Table
- T lines, each containing a 5-tuple with values listed in the same order as described above. The 5-tuple will be wrapped with curly braces ({ and }) and each value will be separated by a comma and a space.
    - Machine states ($q_{current}$ and q_{new}) will always be represented by uppercase letters.
    - Symbols (s_{read} and s_{write}) will always be single-digit non-negative integers.
    - The direction d will always be a capital L (for left) or R (for right).
- A line containing symbols separated by spaces, representing the initial tape given to the machine. The tape may contain any odd number of symbols greater than 1.
- A line containing a positive integer, X, indicating the number of steps to simulate for this machine.

```
2
4
{A, 0, 1, R, A}
{A, 1, 1, L, B}
{B, 1, 0, L, B}
{B, 0, 0, R, A}
0 1 1 0 0 0 1
8
8
{A, 0, 1, R, A}
{A, 1, 2, R, A}
{A, 2, 3, R, A}
{A, 3, 0, L, B}
{B, 0, 3, L, B}
{B, 1, 0, L, B}
{B, 2, 1, L, B}
{B, 3, 2, R, A}
3 3 3 3 3 3 3 3 3 0 3 3 3 3 3 3 3 3 3
20
```

Sample Output

For each test case, your program must print the state of the tape after X steps have been completed by the machine. Print the entire tape on a single line, and separate values on the tape with spaces.

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