## Task

A program that should simulate a Turing machine. Read the input file given in a specific format and tell the route taken by the Turing machine for accepting or rejecting the input string.

## Turing Machine

We have seen FSM and PDAs which take only an input string and a stack as input/data structure, in case of the Turing machine, we have a data structure we call as tape. On the input tape we have a control called head which can move in either direction left or right depending upon the input and transition (production rule). The head can also write on the tape. If the head finishes reading the input string and the machine current state is qA (Accept state), we say that the given input is accepted by the Turing machine and if the current state is qR (Reject state), we say that the given input is rejected by the Turing machine. The accept and reject cases are defined but in case of Turing machine we have might have a case where the machine cannot solve the problem, we call such problem non-halting problems as in such cases machine cannot come to a halt and will continue to run in a "loop".

## Input File

The structure of the input file is as follows:

| **Line numbers** | **Representation** |
|---|---|
| 1 | Number of variables in input alphabet |
| 2 | Number of variables in tape alphabet |
| 3 | Number of states |
| 4 | States |
| 5 | Start state |
| 6 | Accept state |
| 7 | Reject state |
| 8 | Blank symbol |
| 9 | Tape alphabets |
| 10 | Input alphabet |
| 11-25 | Transitions |
| 26-27 | Input strings |

The transition of the form "q1 0 b R q2" means go to q2 when the current state is q2 upon input 0, write blank and goes to Right. So, the first element is current state, second element is input, third symbol is what to be written on the tape, forth one is in which direction to go and then fifth element is the state in which it'll go.

## Implementation

The program is implemented in two classes. File Reader and SimulateTM.

FileReader class as the name suggests is used to read the input text file, the format of which is explained above. The private data types of the class store the read information in the relevant variable. There are a few helper methods such as *setArrayListStr* and *setArrayListInt.* These methods take 2 parameters an

ArrayList and a string. The method converts the string into elements of the ArrayList for either string or integers. Another helper method is *generateTuplesForTT* which takes an ArrayList as parameter and creates a tuple for the transition table as in the given format in input file. Using these helper methods, the FileReader class reads the input file when the constructor is invoked.

The second class SimulateTM simulates a Turing machine. The class has 3 methods besides a parameterized constructor and a main method which is used to invoke classes and call the *simulate* method. The parameterized constructor invokes the FileReader class so that the input file is read. Then, it stores the transition tuples into *tTable* a List of ArrayList and similarly for *inputs*. The main function here is *simulate* which simulates Turing machine. The method creates an output file and reads the first input and a blank symbol in the beginning of the input and at the end. The program then runs until an accept or reject state is found. It then starts with finding the next transition by calling getTransition function. from the current state (always a start state in the beginning of a program). If a transition doesn't exist, the method returns a "NULL" and program comes to halt with a console message of "No transition found.". Then it sets the next state to the current state and the current state to the path taken and move the head on the tape according to the transition to the right or left by calling the method called *moveHead* and set the index of the head to that value.

If an accept or reject state is found, the control then moves out of the loop and prints the route taken for the input string as well as if it is accepted or rejected to console and to an output file. If however the loop keeps running 3 times (a random value) the size of input string, we assume it to be the halting problem and program still ends with a console message of "Halting problem" for that input.

The input and output samples are attached below.

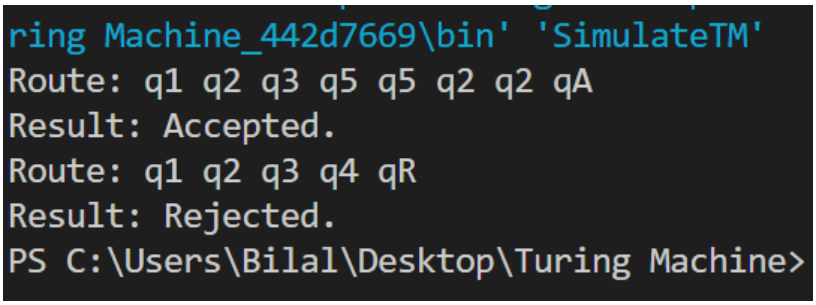


*Figure 1 Console Output*

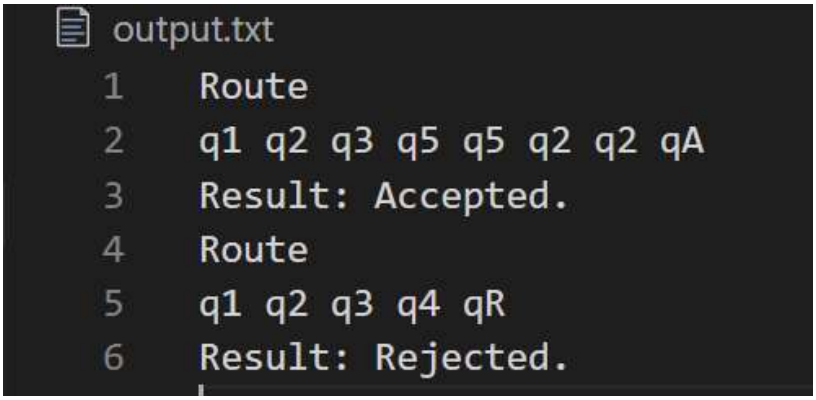


*Figure 2 File Output*

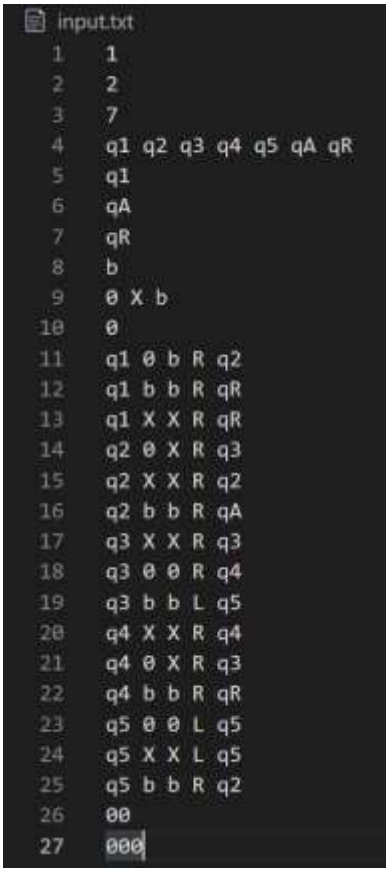


*Figure 3 Input File*

## Note:

Please rename the java file to “SimulateTM.java” and input file to “input.txt” before running otherwise error will occur.