In this project we will simulate a Turing Machine with C++.

First, we need to create an input file. The input file we created is different version of the input file that is given in the document. Therefore, the order of the given inputs are the same.

```
1
0
2
0 X
b
7
q1 q2 q3 q4 q5 qA qR
q1
qA
qR
q1 0 b R q2
q1 b b R qR
q1 X X R qR
q2 0 X R q3
q2 X X R q2
q2 b b R qA
q3 X X R q3
q3 0 0 R q4
q3 b b L q5
q4 X X R q4
q4 0 X R q3
q4 b b R qR
q5 0 0 L q5
q5 X X L q5
q5 b b R q2
000
```

Now we can start to write the code.

We will start coding by creating a class named Data and we collect everything we read from file input in this class's attributes.

```
class Data{
public:

int inputAlphVarNum;
string inputAlph;
int tapeAlphVarNum;
string tapeAlph;
string blank;
int numStates;
string allStates;
string startState;
string acceptState;
string rejectState;
vector<string> states;
string detectedStr;

};
```

Secondly, we need to read input file and collect all data that is given with a class. To do this we create a function named "readInputFile". In this function we take an Data object as a parameter and line by line we assign the values we read from the file.

```
void readInputFile(Data& data){

int counter=0;
string line;

ifstream MyReadFile("input.txt");

while(getline (MyReadFile, line)){

    if(counter == 0){
    data.inputAlphVarNum = stoi(line);
    }else if(counter == 1){
        data.inputAlph = line;
    }else if(counter == 2){
        data.tapeAlphVarNum = stoi(line);
    }else if(counter == 3){
        data.tapeAlph = line;
    }else if(counter == 4){
        data.blank = line;
    }else if(counter == 5){
        data.numStates = stoi(line);
    }else if(counter == 6){
        data.allStates = line;
    }else if(counter == 7){
        data.startState = line;
    }else if(counter == 8){
        data.acceptState = line;
    }else if(counter == 9){
        data.rejectState = line;
    }else {
        data.states.push_back(line);
    }

    counter++;
}
    data.detectedStr = data.states.back();
    data.states.pop_back();
}
```

In the last condition we collect the every state process in a vector but by doing this we also add string to be detected to the vector. Therefore, we need to assign it to it's original value and pop it from the vector.

Now, to start our simulation we will create a function named "simulateTuringMachine". Simulation will be done in this function. As parameter we will send "detected string" which is basically the tape and the object we called "data".

```
simulateTuringMachine(data.detectedStr, data);
```

```
string simulateTuringMachine(string input, Data machine)
{
```

In the beginning we need to divide the input string to chars and assign them into an array. This array will have two more variables which are blank states. One of them will be at the end of the array the other one will be on the beginning of the array.

```
char tape[input.length() + 2];

for (int i = 1; i < input.length() + 1; i++)
{
    tape[i] = input[i];
}
tape[0] = machine.blank;
tape[input.length() + 1] = machine.blank;
```

The reason of these blank variables are to see clearly the beginning and the end of the array.

Then, we initialize the currentState string and visitedStates vector.

```
string currentState = machine.startState;

vector<string> visitedStates;
visitedStates.push_back("ROUT: ");
```

Now in a while loop we can start to do transitions as long as the current state is not accept state and not the reject state. In every loop we will push back the current state to visited states.

```
while (currentState != machine.acceptState && currentState != machine.rejectState)
{

    visitedStates.push_back(currentState);
```

Finally, we can loop through our states and simulate our turing machine. In state lines we know exact position of the inputs therefore we can use static values.

```
for (auto i = machine.states.begin(); i < machine.states.end(); i++)
{
    if ((*i).substr(0, 2) == currentState && (*i)[3] == tape[head])
    {
        tape[head] = (*i)[5];

        if ((*i)[7] == 'R' && tape[head + 1] != 'b')
        {
            head++;
            currentState = (*i).substr(9, 11);
        }
        else if ((*i)[7] == 'R' && tape[head + 1] == 'b')
        {
            currentState = (*i).substr(9, 11);
        }
        else if ((*i)[7] == 'L' && tape[head - 1] != 'b')
        {
            head--;
            currentState = (*i).substr(9, 11);
        }
        else if ((*i)[7] == 'L' && tape[head - 1] == 'b')
        {
            currentState = (*i).substr(9, 11);
        }
    }
}
```

Here in first “if” condition we check if the current state and the value we read is same with the value that heading points to. If it is true we change the value on the tape. Then we check our direction to move from state and we check if next value is blank or not. If it is blank we don’t change the position but we change current state. If it is not blank we change current state and also the head value which points to the value on the tape.

```
string result;

for (auto i = visitedStates.begin(); i < visitedStates.end(); i++)
{
    result = result + " " + *i;
}

if (currentState == machine.acceptState)
{
    result += "\nRESULT: accepted";
}
else if (currentState == machine.rejectState)
{
    result += "\nRESULT: rejected";
}
else
{
    result += "\nRESULT: looped";
}

return result;
```

After this in the same function we create a result string and we assign values of “visitedStates” to it and we indicate if its accepted, rejected or looped.

```
int main()
{

    Data data;

    readInputFile(data);

    string result = simulateTuringMachine(data.detectedStr, data);
    cout << result << endl;

    return 0;
}
```

Finally we print the result.

```
 ROUT:  q1
RESULT: looped
```