# PRACTICAL - 2

**Practical Definition:** String Validation Using Finite Automata

**Objective:** To implement a program that validates a given string against rules defined in terms of finite automata.

## Input requirement:

- Accept rules in the form of finite automata (e.g., states, transitions, start state, accept states) as input.
- Accept a string to be validated against the provided finite automata rules.

#### **Expected output:**

- If the string adheres to the rules of the finite automata, the program should output: "Valid String".
- If the string does not adhere to the rules, the program should output: "Invalid String".

#### **CODE:**

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
    int inputSymbol;
    cout << "No of Input Symbol : ";
    cin >> inputSymbol;

    char array[inputSymbol];
    for (int i = 0; i < inputSymbol; i++) {
        cin >> array[i];
    }

    int states;
    cout << "No of states : ";
    cin >> states;
```

```
int initial;
cout << "Initial state : ";</pre>
cin >> initial;
int finals;
cout << "No Final state : ";</pre>
cin >> finals;
int finalStates[finals];
for (int i = 0; i < finals; i++) {
  cout << "Final state " << i + 1 << ": ";
  cin >> finalStates[i];
}
int transitionTable[states][inputSymbol];
for (int i = 0; i < states; i++) {
  for (int j = 0; j < inputSymbol; j++) {
     cout << "Transition from state " << i + 1 << " on input " << array[j] << " is : ";
     cin >> transitionTable[i][j];
}
cout << "Transition Table : " << endl;</pre>
for (int i = 0; i < states; i++) {
  for (int j = 0; j < inputSymbol; j++) {
     cout << transitionTable[i][j] << " ";</pre>
  }
  cout << endl;
}
```

```
string s;
cout << "Enter String: ";</pre>
cin >> s;
int currentState = initial;
for (char c:s) {
  int inputIndex = -1;
  for (int i = 0; i < inputSymbol; i++) {
     if (array[i] == c) {
        inputIndex = i;
        break;
  if (inputIndex == -1) {
     cout << "Invalid input symbol: " << c << endl;</pre>
     return 1;
  currentState = transitionTable[currentState - 1][inputIndex];
}
bool isAccepted = false;
for (int i = 0; i < finals; i++) {
  if (currentState == finalStates[i]) {
     isAccepted = true;
     break;
if (isAccepted) {
  cout << "String is accepted" << endl;</pre>
```

```
} else {
    cout << "String is not accepted" << endl;
}
return 0;
}</pre>
```

### **OUTPUT:**

```
PS E:\Collage DEPSTAR\SEM-6\Design of Language Processor\Practical> cd "e:\Collage DEPSTAR\SEM-6\Design of Language Processor\Practical\SEM-6\Design of Language Processor\Practical\SEM-6\Design of Language Processor\Practical\SEM-6\Design of Language P
  age Processor\Practical\P2\" ; if ($?) { g++ p2.cpp -o p2 } ; if ($?) { .\p2 }
No of Input Symbol : 2
  ab
   No of states : 4
   Initial state : 1
   No Final state : 1
   Final state 1: 2
   Transition from state 1 on input a is : 2
   Transition from state 1 on input b is : 3
   Transition from state 2 on input a is : \mathbf{1}
   Transition from state 2 on input b is : 4
   Transition from state 3 on input a is : 4
   Transition from state 3 on input b is : 1
   Transition from state 4 on input a is : 3
   Transition from state 4 on input b is : 2
   Transition Table :
   Enter String: abbabab
   String is accepted
```