Faculty of Technology and Engineering

Computer Science and Engineering

Practical

Academic Year	:	2025-26	Semester	:	6
Course code	:	CSE312	Course name	:	Design of language processing

Practical - 2

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

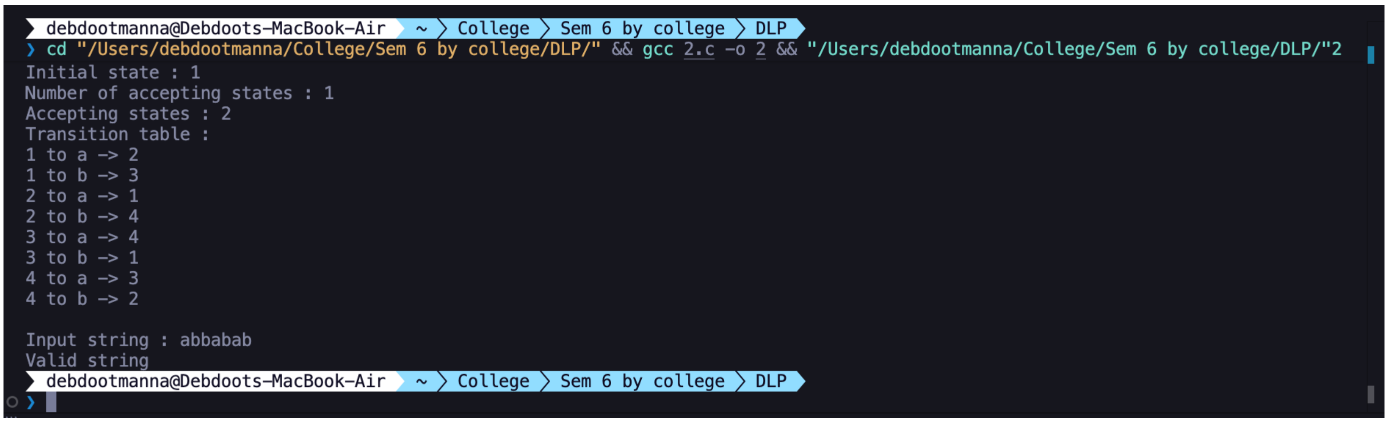
2. Program Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

#define MAX_STATES 100
#define MAX_SYMBOLS 26
#define MAX_STRING_LENGTH 1000

// Structure to represent a Finite Automata
typedef struct {
   int numStates;
   int numSymbols;
   char symbols[MAX_SYMBOLS];
   int startState;
   int numAcceptStates;
   int acceptStates[MAX_STATES];
   int transitionTable[MAX_STATES][MAX_SYMBOLS];
} FiniteAutomata;

int getSymbolIndex(FiniteAutomata *fa, char symbol) {
   for (int i = 0; i < fa->numSymbols; i++) {
      if (fa->symbols[i] == symbol) {
         return i;
      }
   }
   return -1;
```

```
}

bool isAcceptState(FiniteAutomata *fa, int state) {
    for (int i = 0; i < fa->numAcceptStates; i++) {
        if (fa->acceptStates[i] == state) {
            return true;
        }
    }
    return false;
}

bool validateString(FiniteAutomata *fa, char *inputString) {
    int currentState = fa->startState;
    int len = strlen(inputString);


    for (int i = 0; i < len; i++) {
        char currentChar = inputString[i];
        int symbolIndex = getSymbolIndex(fa, currentChar);
                // If symbol is not in the alphabet, reject the string
        if (symbolIndex == -1) {
            return false;
        }

        int nextState = fa->transitionTable[currentState][symbolIndex];

        if (nextState == -1) {
            return false;
        }

        currentState = nextState;
    }

    // Accept if final state is an accepting state
    return isAcceptState(fa, currentState);
}

int main() {
    FiniteAutomata fa;

    // Initialize transition table with -1 (representing no transition)
    for (int i = 0; i < MAX_STATES; i++) {
        for (int j = 0; j < MAX_SYMBOLS; j++) {
            fa.transitionTable[i][j] = -1;
        }
    }

    printf("Number of input symbols : ");
    scanf("%d", &fa.numSymbols);

    // Input: The symbols themselves
```

```
  printf("Input symbols : ");
  for (int i = 0; i < fa.numSymbols; i++) {
    scanf(" %c", &fa.symbols[i]);
  }

  printf("Enter number of states : ");
  scanf("%d", &fa.numStates);

  printf("Initial state : ");
  scanf("%d", &fa.startState);

  printf("Number of accepting states : ");
  scanf("%d", &fa.numAcceptStates);

  printf("Accepting states : ");
  for (int i = 0; i < fa.numAcceptStates; i++) {
    scanf("%d", &fa.acceptStates[i]);
  }

  // Input: Transition table
  printf("Transition table :\n");

  // Read all transitions (state x symbol combinations)
  int totalTransitions = fa.numStates * fa.numSymbols;

  for (int i = 0; i < totalTransitions; i++) {
    int fromState, toState;
    char symbol;

    scanf("%d to %c -> %d", &fromState, &symbol, &toState);

    int symbolIndex = getSymbolIndex(&fa, symbol);
    if (symbolIndex != -1) {
      fa.transitionTable[fromState][symbolIndex] = toState;
    }
  }

  char inputString[MAX_STRING_LENGTH];
  printf("\nInput string : ");
  scanf("%s", inputString);

  if (validateString(&fa, inputString)) {
    printf("Valid string\n");
  } else {
    printf("Invalid string\n");
  }

  return 0;
}
```

3.Output:



```
 debdootmanna@Debdoots-MacBook-Air  ~  College  Sem 6 by college  DLP
❯ cd "/Users/debdootmanna/College/Sem 6 by college/DLP/" && gcc 2.c -o 2 && "/Users/debdootmanna/College/Sem 6 by college/DLP/"2
Initial state : 1
Number of accepting states : 1
Accepting states : 2
Transition table :
1 to a -> 2
1 to b -> 3
2 to a -> 1
2 to b -> 4
3 to a -> 4
3 to b -> 1
4 to a -> 3
4 to b -> 2

Input string : abbabab
Valid string
 debdootmanna@Debdoots-MacBook-Air  ~  College  Sem 6 by college  DLP
❯
```