LAB EXERCISE – 1

(1) /* Find prefix, suffix and substring from given string */

```
#include <iostream>
#include <string.h>
#include <set>
using namespace std;

int main()
{
   char str[100], prefix[100], suffix[100], substring[100];
   int i, j, k, l, m, n, o, p;

   cout << "Enter a string: ";
   cin >> str;
   l = strlen(str);

   cout << "Prefix: ";
   for (i = 0; i < l; i++)
   {
      for (j = 0; j <= i; j++)
      {
         prefix[j] = str[j];
      }
      prefix[j] = '\0';
```

```
    cout << prefix << " ";
}
cout << endl;

cout << "Suffix: ";
for (k = 0; k < l; k++)
{
    for (m = k; m < l; m++)
    {
        suffix[m - k] = str[m];
    }
    suffix[m - k] = '\0';
    cout << suffix << " ";
}
cout << endl;

cout << "Substring: ";
set<string> unique_substrings; // Use a set to keep track of unique substrings
for (n = 0; n < l; n++)
{
    for (o = n; o < l; o++)
    {
        // Extract substring using string.substr() function
        string sub = string(str).substr(n, o - n + 1); // (start, length) of substring
        // Check if substring has already been generated
        if (unique_substrings.find(sub) == unique_substrings.end())
```

```
            {
                unique_substrings.insert(sub);
                cout << sub << " ";
            }
        }
    }
    cout << endl;

    return 0;
}
```

(2) /* DFA for the language of string over {a,b} such that each string contains "aba" as substring */

```
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string str;     // string to be checked
    char state = 0; // initial state (q0)

    cout << "Enter the string: ";
    cin >> str;
```

```
// loop to check each character of the string for the DFA
for (int i = 0; i < str.length(); i++)
{
   // check if the string is over {a,b} or not
   if (str[i] != 'a' && str[i] != 'b')
   {
      cout << "String not accepted.\nPlease enter a string over {a,b}" << endl;
      return 0;
   }

   // dfa transition check
   if (state == 0 && str[i] == 'a')
      state = 1;
   else if (state == 0 && str[i] == 'b')
      state = 0;
   else if (state == 1 && str[i] == 'a')
      state = 1;
   else if (state == 1 && str[i] == 'b')
      state = 2;
   else if (state == 2 && str[i] == 'a')
      state = 3;
   else if (state == 2 && str[i] == 'b')
      state = 0;
   else if (state == 3 && str[i] == 'a')
      state = 3;
   else if (state == 3 && str[i] == 'b')
```

```
        state = 3;
    }

    // check if the string is accepted or not,
    // i.e. if the final state is 3 then string is accepted
    // else string is not accepted
    if (state == 3)
        cout << "String accepted";
    else
        cout << "String not accepted";

    return 0;
}
```

(2) /* NFA for the language of string over {0,1} such that each string starts with "10" */

```
#include <iostream>
#include <vector>
using namespace std;

// Define the NFA as a set of states and transitions
vector<int> states = {0, 1, 2}; // States are represented by integers (0, 1, 2, 3, ...)
vector<vector<pair<char, int>>> transitions = {
    {{'1', 1}},
    {{'0', 2}},
```

```
    {{'0', 2}, {'1', 2}},
    {{}}};
// Transitions are represented by pairs of characters and states (character, state)

// Define a function to simulate the NFA on a given string
bool simulate_nfa(string input)
{
    // Start at the initial state (state 0)
    vector<int> current_states = {0};

    // Loop through each character in the input string
    for (char c : input)
    {
        //Find all possible transitions from the current states for the current character
        vector<int> next_states;
        for (int state : current_states)
        {
            for (auto transition : transitions[state])
            {
                if (transition.first == c)
                {
                    next_states.push_back(transition.second);
                }
            }
        }
        // If there are no possible transitions, the input string is not accepted
```

```
        if (next_states.empty())
        {
            return false;
        }
        // Update the current states to the next states
        current_states = next_states;
    }
    // If the final state is an accepting state, the input string is accepted
    for (int state : current_states)
    {
        if (state == 2)
        {
            return true;
        }
    }
    return false;
}

// Define the main function to run the program
int main()
{
    // Get input from the user
    string input;
    cout << "Enter a string to check: ";
    cin >> input;
```

```
    // Simulate the NFA on the input string and output the result
    if (simulate_nfa(input))
    {
        cout << "String starts with 10." << endl;
    }
    else
    {
        cout << "String does not start with 10." << endl;
    }

    return 0;
}
```