

LAB -1

WAP to Generate Prefix, Suffix, and Substring Combinations of a String.

Theory:

This program allows a user to generate combinations of a given string in three forms: prefixes, suffixes, and substrings. A prefix is the starting sequence of characters, a suffix is the ending sequence, and substrings are any sequence of consecutive characters within the string. The user inputs a string and selects an option to view the desired combination. Based on the selection, the program loops through the string, displaying the appropriate combinations.

Program:

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

void prefix(const string&);
void suffix(const string&);
void substring(const string&);

int main() {
    string str;
    int choice;

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

    do {
        cout << "\n1. Prefix Combination\n2. Suffix Combination\n3. Substring Combination\n4. Other Number to exit\nEnter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                prefix(str);
                break;
            case 2:
                suffix(str);
                break;
            case 3:
                substring(str);
                break;
            default:
                cout << "Exiting program!" << endl;
                choice = 0;
                break;
        }
    } while (choice != 0);
}
```

```

    return 0;
}

void prefix(const string& s) {
    cout << "\nPrefix Combinations are:- " << endl;
    string temp;
    for (int i = 0; i < s.length(); i++) {
        temp += s[i];
        cout << temp << endl;
    }
    cout << endl;
}

void suffix(const string& s) {
    cout << "\nSuffix Combinations are:- " << endl;
    string temp;
    for (int i = s.length() - 1; i >= 0; i--) {
        temp = s[i] + temp;
        cout << temp << endl;
    }
    cout << endl;
}

void substring(const string& s) {
    cout << "\nSubstring Combinations are:- " << endl;
    for (int i = 0; i < s.length(); i++) {
        string temp;
        for (int j = i; j < s.length(); j++) {
            temp += s[j];
            cout << temp << endl;
        }
    }
    cout << endl;
}

```

Conclusion:

The program provides an interactive method to generate prefix, suffix, and substring combinations from a user-provided string, demonstrating basic string manipulation in C++.

Enter the string: abcd

1. Prefix Combination
2. Suffix Combination
3. Substring Combination
4. Other Number to exit

Enter your choice: 1

Prefix Combinations are:-

a
ab
abc
abcd

1. Prefix Combination
2. Suffix Combination
3. Substring Combination
4. Other Number to exit

Enter your choice: 2

Suffix Combinations are:-

d
cd
bcd
abcd

1. Prefix Combination
2. Suffix Combination
3. Substring Combination
4. Other Number to exit

Enter your choice: 3

Substring Combinations are:-

a
ab
abc
abcd
b
bc
bcd
c
cd
d

LAB-2

DFA Program to Accept Strings Starting and Ending with 'a' or 'b' Over $\Sigma = \{a, b\}$.

Theory:

In automata theory, a Deterministic Finite Automaton (DFA) is a machine used to recognize specific patterns within input strings. It consists of a finite number of states, including a start state, accept states, and transition rules that describe how to move from one state to another based on the current input character.

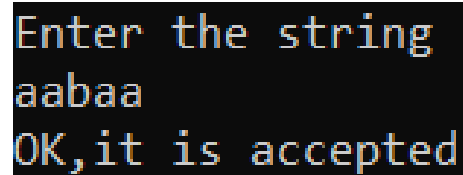
This DFA-based program is designed to accept strings that begin with either the character 'a' or 'b' and end with the same character (i.e., strings that either start and end with 'a' or start and end with 'b'). The program checks the input string step by step using transitions between various states. If the input satisfies the conditions of the DFA, the string is accepted; otherwise, it is rejected.

PROGRAM

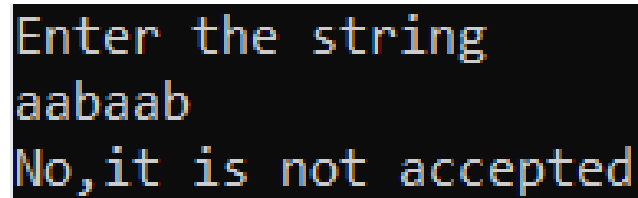
```
#include<iostream>
using namespace std;

void q1(string, int);
void q2(string, int);
void q3(string, int);
void q4(string, int);
void q0(string s, int i) {
    if(i == s.length()) cout << "No, it is not accepted\n";
    else if(s[i] == 'a') q1(s, i+1);
    else q3(s, i+1);
}
void q1(string s, int i) {
    if(i == s.length()) cout << "OK, it is accepted\n";
    else if(s[i] == 'a') q1(s, i+1);
    else q2(s, i+1);
}
void q2(string s, int i) {
    if(i == s.length()) cout << "No, it is not accepted\n";
    else if(s[i] == 'a') q1(s, i+1);
    else q2(s, i+1);
}
void q3(string s, int i) {
    if(i == s.length()) cout << "OK, it is accepted\n";
    else if(s[i] == 'a') q4(s, i+1);
    else q3(s, i+1);
}
void q4(string s, int i) {
    if(i == s.length()) cout << "No, it is not accepted\n";
```

```
    else if(s[i] == 'a') q4(s, i+1);  
    else q3(s, i+1);  
}  
int main() {  
    string s;  
    cout << "Enter the string\n";  
    cin >> s;  
    q0(s, 0);  
}
```



```
Enter the string  
aabaa  
OK,it is accepted
```



```
Enter the string  
aabaab  
No,it is not accepted
```

Conclusion:

This DFA successfully processes strings over the alphabet {a, b}, accepting strings that begin and end with the same character. It implements five states and transitions based on input characters to decide acceptance or rejection.

LAB-3

WAP in any high level language design a DFA with $\Sigma=\{0,1\}$ that accept the language L ending with 01.

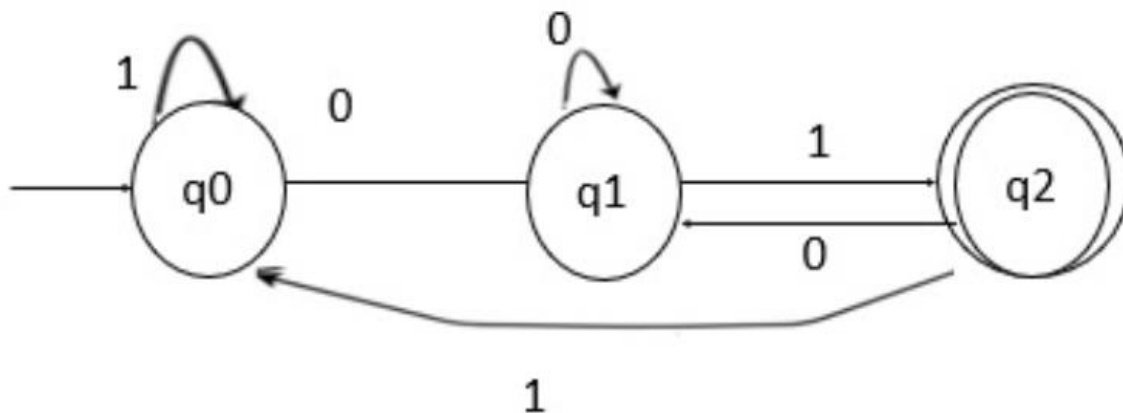
Theory

This program implements a Deterministic Finite Automaton (DFA) to recognize binary strings over the alphabet $\Sigma=\{0,1\}$ that end with the substring "01". The DFA consists of three states:

- State 'a': The initial state, where no valid ending has been encountered.
- State 'b': Indicates that the last character read was '0'.
- State 'c': Indicates that the last two characters read were "01", marking an accepting state.

The DFA transitions between these states based on the input characters, allowing it to validate whether the input string conforms to the defined language.

Transition Diagram:



PROGRAM:

```
#include <stdio.h>
#define max 100
main()
{
char str[max],f='a';
int i;
printf("enter the string to be checked: ");
scanf("%s",str);
for(i=0;str[i]!='\0';i++)
{
switch(f)
{
case 'a': if(str[i]=='0') f='b';
else if(str[i]=='1') f='a';
break;
case 'b': if(str[i]=='0') f='b';
else if(str[i]=='1') f='c';
break;
case 'c': if(str[i]=='0') f='b';
else if(str[i]=='1') f='a';
break;
}
}
if(f=='c')
printf("\nString is accepted", f);
else printf("\nString is not accepted", f);
return 0;
}
```

OUTPUT:

```
enter the string to be checked: 0001
```

```
String is accepted
```

```
-----
```

```
enter the string to be checked: 11011
```

```
String is not accepted
```

```
-----
```

Conclusion:

The implemented DFA effectively determines whether a binary string ends with "01". By utilizing state transitions based on input characters, the program efficiently checks the acceptance condition.

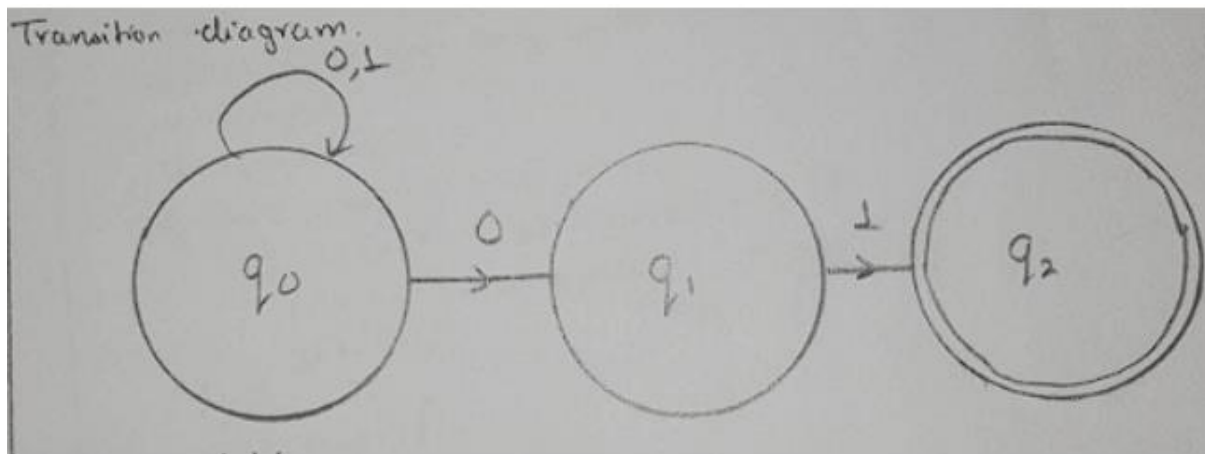
LAB 4:

WAP in any high level language design a NFA with $\Sigma=\{0,1\}$ that accept the language L ending with 01.

Theory:

This program implements a Non-deterministic Finite Automaton (NFA) to check if a given binary string ends with the substring "01". An NFA is a theoretical model of computation that can be in multiple states at once. In this program, the NFA transitions between states based on input characters ('0' and '1'). The user inputs a binary string, and the program simulates the NFA to determine if the string meets the criteria.

Transition Diagram:



PROGRAM

```
#include <iostream>
#include <vector>
#include <string>

using namespace std;
```

```

vector<int> states = {0, 1, 2};
vector<vector<pair<char, int>>> transitions = {
    {{'0', 0}, {'1', 0}, {'0', 1}},
    {{'1', 2}},
    {{}}};

bool simulate_nfa(string input)
{
    vector<int> current_states = {0};

    for (char c : input)
    {
        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 (next_states.empty())
        {
            return false;
        }
        current_states = next_states;
    }

    for (int state : current_states)
    {
        if (state == 2)
        {
            return true;
        }
    }
    return false;
}

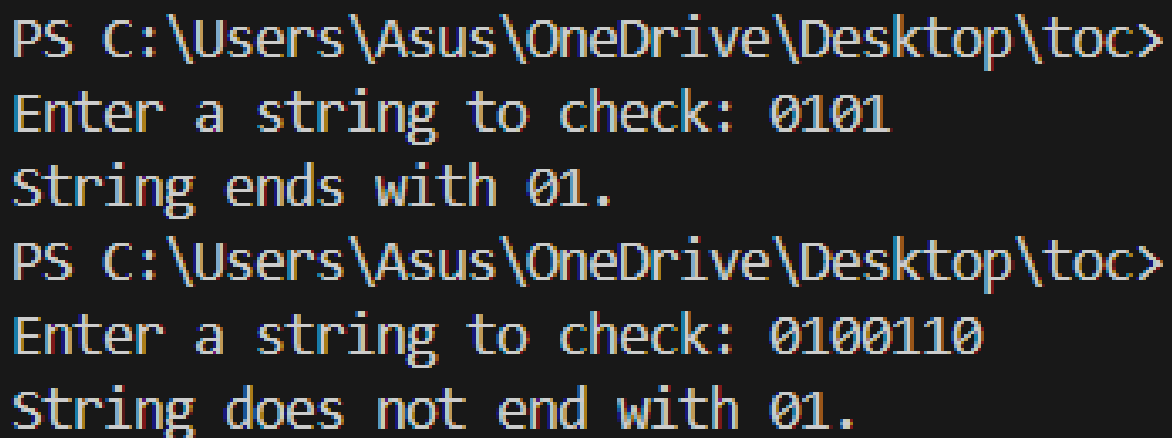
int main()
{

```

```
string input;
cout << "Enter a binary string to check: ";
cin >> input;

if (simulate_nfa(input))
{
    cout << "String ends with '01'." << endl;
}
else
{
    cout << "String does not end with '01'." << endl;
}

return 0;
}
```



```
PS C:\Users\Asus\OneDrive\Desktop\toc>
Enter a string to check: 0101
String ends with 01.
PS C:\Users\Asus\OneDrive\Desktop\toc>
Enter a string to check: 0100110
String does not end with 01.
```

Conclusion:

The program effectively simulates a Non-deterministic Finite Automaton to check if a user-provided binary string ends with the substring "01".

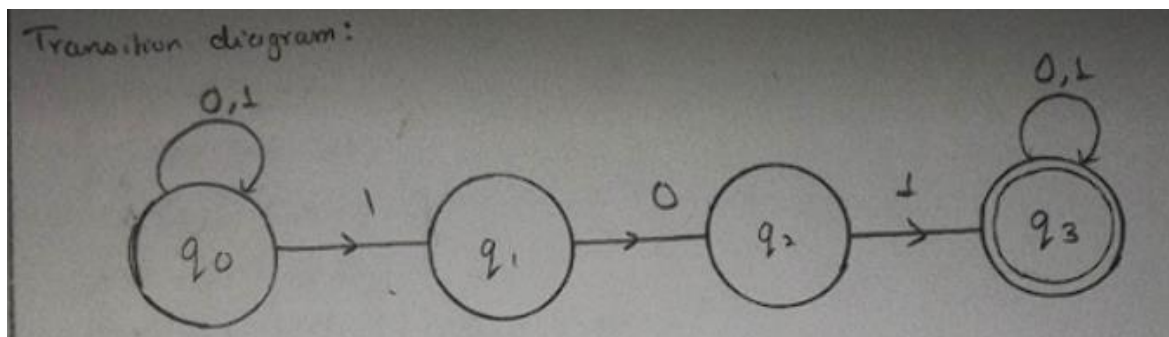
Lab-5:

WAP in any high level language design a NFA with $\Sigma=\{0,1\}$ that accept the language L ending with 101.

Theory:

This program implements a Non-deterministic Finite Automaton (NFA) to check if a given binary string contains the substring "101". An NFA allows for multiple possible transitions for each input character. The user inputs a binary string, and the program simulates the NFA to determine if the string includes the desired substring.

Transition Diagram:



Program:

```
#include <iostream>
#include <vector>
#include <string>
```

```
using namespace std;
```

```
vector<int> states = {0, 1, 2, 3};
vector<vector<pair<char, int>>> transitions = {
```

```
{{'0', 0}, {'1', 0}, {'1', 1}},  
{{'0', 2}},  
{{'1', 3}},  
{{'0', 3}, {'1', 3}}};
```

```
bool simulate_nfa(string input)  
{  
    vector<int> current_states = {0};  
  
    for (char c : input)  
    {  
        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 (next_states.empty())  
        {  
            return false;  
        }  
        current_states = next_states;  
    }  
  
    for (int state : current_states)  
    {  
        if (state == 3)
```

```
    {  
        return true;  
    }  
}  
return false;  
}
```

```
int main()  
{  
    string input;  
    cout << "Enter a binary string to check: ";  
    cin >> input;  
  
    if (simulate_nfa(input))  
    {  
        cout << "String contains substring '101'." << endl;  
    }  
    else  
    {  
        cout << "String does not contain substring '101'." <<  
endl;  
    }  
  
    return 0;  
}
```

```
PS C:\Users\Asus\OneDrive\Desktop\toc> cd  
Enter a binary string to check: 10101  
String contains substring '101'.  
PS C:\Users\Asus\OneDrive\Desktop\toc> cd  
Enter a binary string to check: 1000100  
String does not contain substring '101'.
```

Conclusion:

The program successfully simulates a Non-deterministic Finite Automaton to check if a user-provided binary string contains the substring "101".