Tribhuvan University
Prithivi Narayan Campus
Pokhara,Kaski

Lab-report of Theory of Computation

Subject code: CSC

Submitted to:
Prithvi Raj Paneru
Department of CSIT
Prithivi Narayan Campus

Submitted by:
Sabin Paudel
Roll no: 56
4th semester

LAB INDEX

Name: Sabin Paudel Roll. No: 56

Level: Bachelors Year/Sem: 2nd/4th

Subject: Theory of Computation Faculty:B.Sc.CSIT
Course No: CSC 262

Lab No	Exp NO	Experiment name	Date of experiment	Submitted Date	Remark
1		Finite Automata			
	1.1	To Simulate Deterministic Finite Automata (DFA)	2081/03/25	2081/04/04	
	1.2	To Simulate Non-Deterministic Finite Automata (NFA)	2081/03/25	2081/04/04	

EXPERIMENT 1.1

TO SIMULATE DETERMINISTIC FINITE AUTOMATA (DFA)

1. OBJECTIVE:

- To study about DFA and simulate DFA using C program.
- Implementation of a DFA for language L = {w | w starts with 01 and consists of 0s and 1s}

2. THEORY:

A DFA is a mathematical model of computation that recognizes patterns within input strings. It is defined by the following components:

- States (Q): A finite set of states.
- Alphabet (Σ): A finite set of input symbols.
- Transition Function (δ): A function that takes a state and an input symbol and returns a new state, denoted as δ: Q × Σ → Q.
- Start State (q0): The state where the DFA begins.
- Accept States (F): A set of states where, if the DFA finishes processing the input, the input is considered accepted by the DFA.
- DFA for the Language L = {w | w starts with 01 and consists of 0s and 1s}

Given the components:

- States: {q0, q1, q2, q3}
- Alphabet: {0, 1}
- Transition Function:
- δ(q0, 0) = q1
- δ(q0, 1) = q3
- δ(q1, 0) = q3
- δ(q1, 1) = q2
- δ(q2, 0) = q2
- δ(q2, 1) = q2
- δ(q3, 0) = q3
- δ(q3, 1) = q3
- Start State: q0
- Accept State: q2

Input String	Output
01	Accept
100	Reject
0010	Reject
011	Accept

3. DEMONSTRATION

Program 1:

To demonstrate the DFA for language L = {w | w starts with 01 and consists of 0s and 1s}

Source code:

```
#include <stdio.h>
#include <stdlib.h>
enum states {q0, q1, q2, q3};
enum states delta(enum states s, char ch) {
    enum states curr_state;
    switch (s) {
        case q0:
            if (ch == '0')
```

```
                curr_state = q1;
            else
                curr_state = q3;
            break;
        case q1:
            if (ch == '1')
                curr_state = q2;
            else
                curr_state = q3;
            break;
        case q2:
            if (ch == '0' || ch == '1')
                curr_state = q2;
            else
                curr_state = q3;
            break;
        case q3:
            curr_state = q3;
            break;
    }
    return curr_state;
}

int main() {
    char input[20];
    enum states curr_state = q0;
    int i = 0;
    printf("Enter the string: ");
    scanf("%s", input);
    char ch = input[i];
    while (ch != '\0') {
        curr_state = delta(curr_state, ch);
        ch = input[++i];
    }
    if (curr_state == q2)
        printf("Accepted\n");
    else
        printf("Rejected\n");
    return 0;
}
```

Program 2:

To demonstrate the DFA for language L = {w | w ends with 01 and consists of 0s and 1s}

Source code:

```
#include <stdio.h>
#include <stdlib.h>
enum states {q0, q1, q2, q3};
enum states delta(enum states s, char ch) {
    switch (s) {
        case q0: return (ch == '0') ? q1 : q0;
        case q1: return (ch == '1') ? q2 : (ch == '0') ? q1 : q0;
        case q2: return (ch == '0') ? q1 : q0;
        default: return q3;
    }
}
int main() {
    char input[20];
    enum states curr_state = q0;
    printf("Enter the string: ");
    scanf("%s", input);
    for (int i = 0; input[i] != '\0'; i++) {
        curr_state = delta(curr_state, input[i]);
    }
    if (curr_state == q2)
        printf("Accepted\n");
    else
        printf("Rejected\n");
    return 0;
}
```

Program 3:

To demonstrate the DFA for language L = {w | has 01 as substring and consists of 0s and 1s}

Source code:

```
#include <stdio.h>
#include <stdlib.h>
enum states {q0, q1, q2};
enum states delta(enum states s, char ch) {
    switch (s) {
        case q0: return (ch == '0') ? q1 : q0;
        case q1: return (ch == '1') ? q2 : (ch == '0') ? q1 : q0;
        case q2: return q2;
        default: return q0;
    }
}
int main() {
    char input[20];
    enum states curr_state = q0;
    printf("Enter the string: ");
    scanf("%s", input);
    for (int i = 0; input[i] != '\0'; i++) {
        curr_state = delta(curr_state, input[i]);
    }
    if (curr_state == q2)
        printf("Accepted\n");
    else
        printf("Rejected\n");
    return 0;
}
```

Program 4:

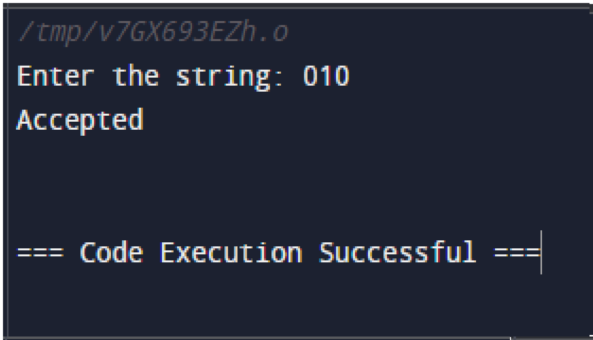
To demonstrate the DFA for language L = {w | w has even no of 0’s and consists of 0s and 1s}

Source code:

```
#include <stdio.h>
#include <stdlib.h>
enum states {q0, q1};
enum states delta(enum states s, char ch) {
    switch (s) {
        case q0: return (ch == '0') ? q1 : q0;
        case q1: return (ch == '0') ? q0 : q1;
        default: return q0;
    }
}
int main() {
    char input[20];
    enum states curr_state = q0;
    printf("Enter the string: ");
    scanf("%s", input);
    for (int i = 0; input[i] != '\0'; i++) {
        curr_state = delta(curr_state, input[i]);
    }
    if (curr_state == q0)
        printf("Accepted\n");
    else
        printf("Rejected\n");
    return 0;
}
```

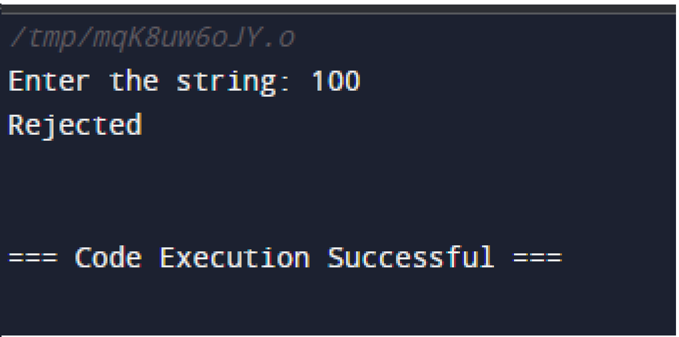
OUTPUT AND DISCUSSION:

Output 1:



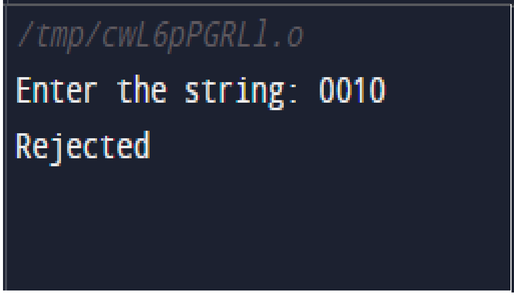
```
/tmp/v7GX693EZh.o
Enter the string: 010
Accepted


=== Code Execution Successful ===
```

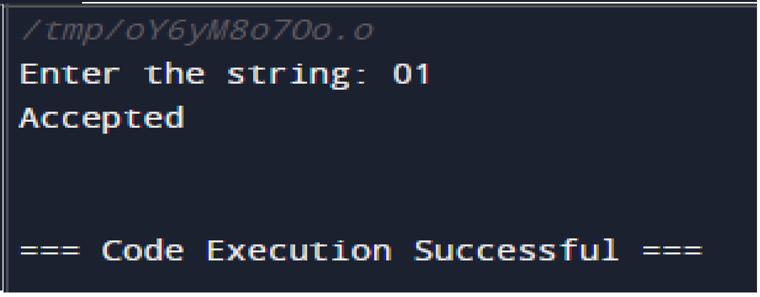


```
/tmp/mqK8uw6oJY.o
Enter the string: 100
Rejected


=== Code Execution Successful ===
```



```
/tmp/cwL6pPGRLl.o
Enter the string: 0010
Rejected
```



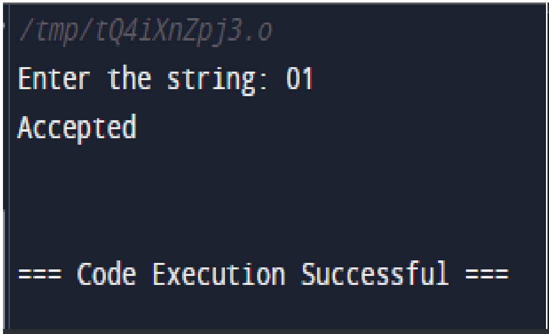
```
/tmp/oY6yM8o7Oo.o
Enter the string: 01
Accepted


=== Code Execution Successful ===
```

Discussion part:

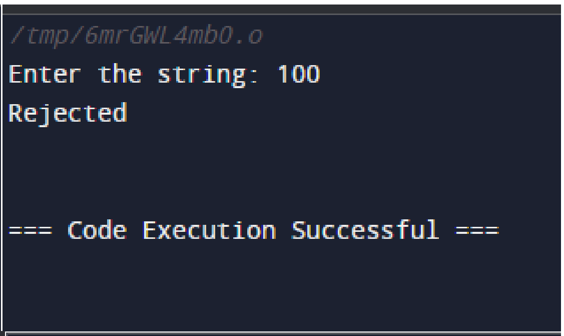
Input	Output	Discussion
01	Accepted	The DFA transitions from q0 to q1 on '0', then from q1 to q2 on '1'. Thus, the string is accepted.
0010	Rejected	The DFA transitions from q0 to q1 on '0', then from q1 to q3 on '0'. Since q3 is not an accept state, the string is rejected.
010	Accepted	The DFA transitions from q0 to q1 on '0', then from q1 to q2 on '1', and stays in q2 on '0'. Thus, the string is accepted
100	Rejected	The DFA transitions from q0 to q3 on '1'. Since q3 is not an accept state, the string is rejected.

Output 2:



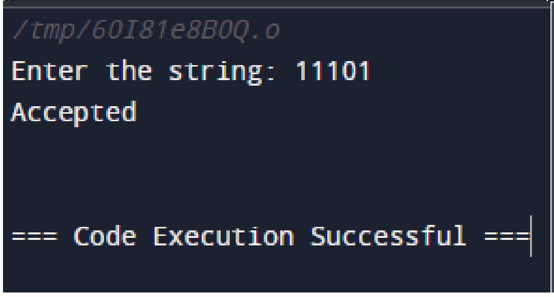
```
/tmp/tQ4iXnZpj3.o
Enter the string: 01
Accepted


=== Code Execution Successful ===
```



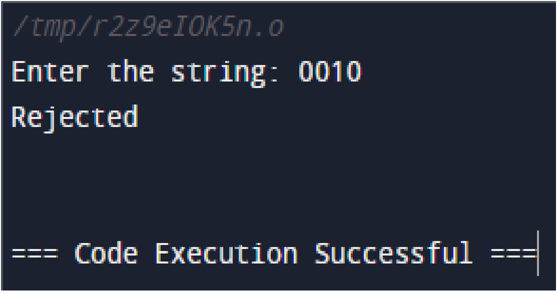
```
/tmp/6mrGWL4mb0.o
Enter the string: 100
Rejected


=== Code Execution Successful ===
```



```
/tmp/6OI81e8B0Q.o
Enter the string: 11101
Accepted


=== Code Execution Successful ===
```



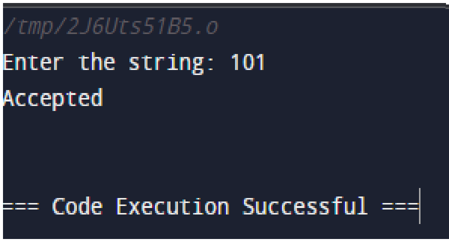
```
/tmp/r2z9eIOK5n.o
Enter the string: 0010
Rejected


=== Code Execution Successful ===
```

Discussion part :

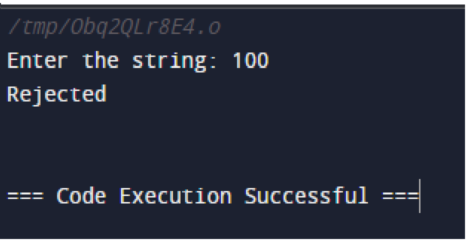
Input	Output	Discussion
01	Accepted	The DFA transitions from q0 to q1 on '0', then from q1 to q2 on '1'. Thus, the string is accepted.
11101	Accepted	The DFA transitions from q0 to q0 on '1', then from q0 to q0 on '1', then from q0 to q1 on '0', then from q1 to q2 on '1'. Thus, the string is accepted.
100	Rejected	The DFA transitions from q0 to q0 on '1', then from q0 to q0 on '0', but there is no transition to an accept state on the final '0'. Thus, the string is rejected.
0010	Rejected	The DFA transitions from q0 to q1 on '0', then from q1 to q1 on '0', then from q1 to q0 on '1'. Thus, the string is rejected.

Output 3:



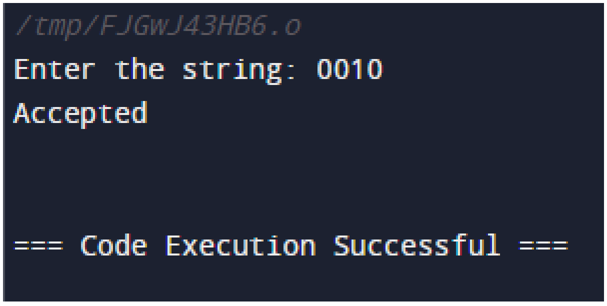
```
/tmp/2J6Uts51B5.o
Enter the string: 101
Accepted


=== Code Execution Successful ===
```



```
/tmp/Obq2QLr8E4.o
Enter the string: 100
Rejected


=== Code Execution Successful ===
```



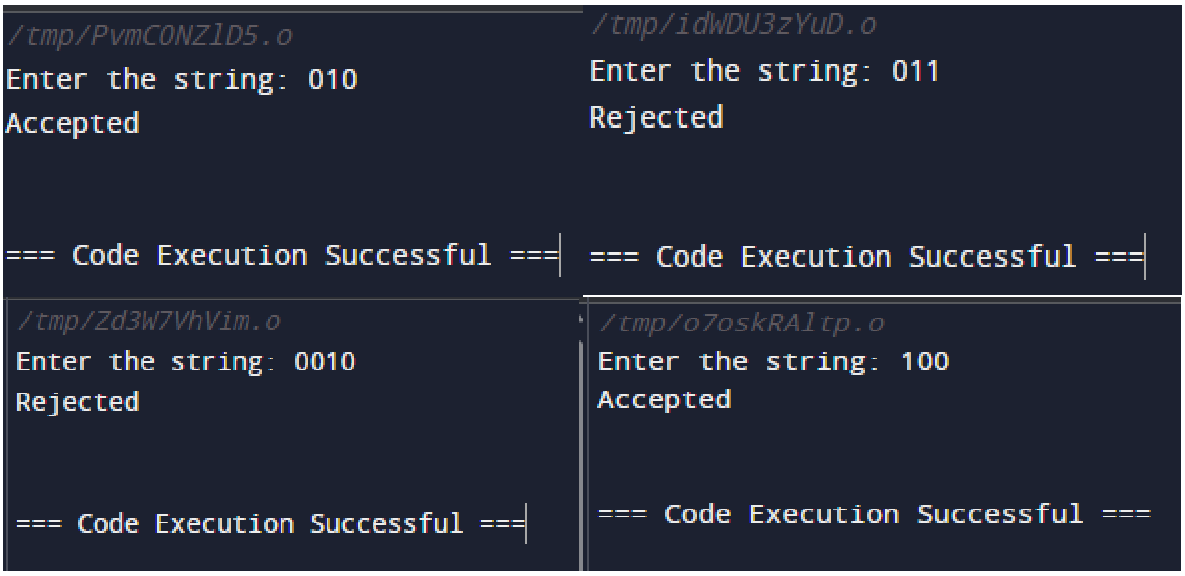
```
/tmp/FJGwJ43HB6.o
Enter the string: 0010
Accepted


=== Code Execution Successful ===
```

Discussion part :

Input	Output	Discussion
0010	Accepted	The DFA transitions from q0 to q1 on '0', then from q1 to q1 on '0', then from q1 to q0 on '1'. Thus, the string is accepted.
101	Accepted	The DFA transitions from q0 to q0 on '1', then from q0 to q1 on '0', then from q1 to q2 on '1'. Thus, the string is accepted.
100	Rejected	The DFA transitions from q0 to q0 on '1', then from q0 to q0 on '0', but there is no transition to an accept state on the final '0'. Thus, the string is rejected.

Output 4:



```
/tmp/PvmCONZ1D5.o
Enter the string: 010
Accepted


=== Code Execution Successful ===
```

```
/tmp/idWDU3zYuD.o
Enter the string: 011
Rejected


=== Code Execution Successful ===
```

```
/tmp/Zd3W7VhVim.o
Enter the string: 0010
Rejected


=== Code Execution Successful ===
```

```
/tmp/o7oskRAltp.o
Enter the string: 100
Accepted


=== Code Execution Successful ===
```

Discussion part :

Input	Output	Discussion
011	Rejected	The DFA transitions from q0 to q1 on '0', then from q1 to q2 on '1', but stays in q2 on the next '1'. Since there is an odd number of 0's, the string is rejected.
0010	Accepted	The DFA transitions from q0 to q1 on '0', then from q1 to q1 on '0', then from q1 to q0 on '1', then from q0 to q1 on '0'. Since there are even number of 0's, the string is accepted.
010	Accepted	The DFA transitions from q0 to q1 on '0', then from q1 to q2 on '1', then from q2 to q1 on '0'. Since there are even number of 0's, the string is accepted.
100	Accepted	The DFA transitions from q0 to q0 on '1', then from q0 to q0 on '0', but there is no transition to an accept state on the final '0'. Thus, the string is accepted.

4. CONCLUSION

The DFA was successfully implemented in C, accurately identifying whether input strings were part of the language based on specified criteria. This experiment provided a thorough understanding of DFA concepts and their practical implementation.

EXPERIMENT 1.2
TO SIMULATE NON-DETERMINISTIC FINITE AUTOMATA (NFA)

1. OBJECTIVE:

- To study about NFA and simulate NFA using C program.
- Implementation of a NFA for language L = {w | w ends with 01 and consists of 0s and 1s}

2. THEORY:

A Nondeterministic Finite Automaton (NFA) is a theoretical model of computation designed to recognize patterns within input strings. Unlike a Deterministic Finite Automaton (DFA), an NFA can transition to multiple states for a given input symbol or even transition without any input (using epsilon transitions). An NFA is composed of the following elements:

- A finite set of states.
- A finite set of input symbols (alphabet).
- A transition function that takes a state and an input symbol and returns a set of possible next states.
- A start state.
- A set of accept states.

3. DEMONSTRATION

Program 1:

To demonstrate NFA for language L = {w | w ends with 01 and consists of 0s and 1s}.

Source code:

```
#include <stdio.h>
#include <stdlib.h>
enum states {q0,q1,q2};
enum states delta(enum states s, char ch){
  enum states curr_state;
  switch (s)
  {
  case q0:
    if (ch == '0')
      curr_state = q1;
    else
      curr_state = q0;
    break;
  case q1:
    if (ch == '1')
      curr_state = q2;
    else
      curr_state = q1;
    break;
  case q2:
    if (ch == '0')
      curr_state = q1;
    else
      curr_state = q0;
```

```
        break;
    }
    return curr_state;
}
int main(){
    char input[20];
    enum states curr_state = q0;
    int i = 0;
    printf("Enter the string:");
    scanf("%s", input);
    char ch = input[i];
    while (ch!='\0'){
        curr_state = delta(curr_state,ch);
        ch = input[++i];
    }
    if (curr_state == q2)
        printf("Accepted");
    else
        printf("Rejected");
    return 0;
}
```

Program 2: Implementing NFA for language L={w | w ends with 11}

Source Code:

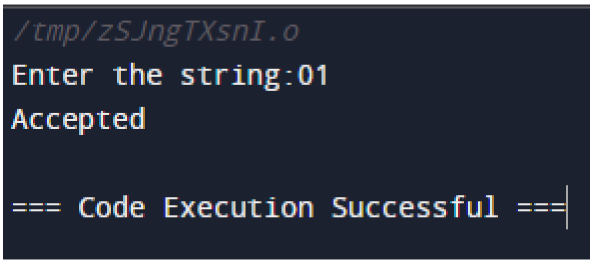
```
#include <stdio.h>
enum states { q0, q1, q2 };
enum states delta(enum states s, char ch) {
    switch (s) {
    case q0:
        if (ch == '1')
            return q1;
        break;
    case q1:
        if (ch == '1')
            return q2;
        break;
    }
    return q0; // Default to q0 (invalid transition)
}
int main() {
    char input[20];
    enum states curr_state = q0;
    int i = 0;
    printf("Enter the string:");
    scanf("%s", input);
    while (input[i] != '\0') {
        curr_state = delta(curr_state, input[i]);
        i++;
    }
    if (curr_state == q2)
        printf("Accepted\n");
    else
        printf("Rejected\n");  return 0;}
```

Program 3: Implementing NFA for language L={w | w containing 01 substring}

Source Code:

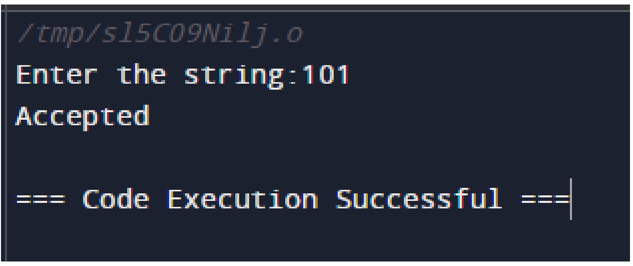
```
#include <stdio.h>
enum states { q0, q1, q2 };
enum states delta(enum states s, char ch) {
    switch (s) {
    case q0:
        if (ch == '0')
            return q1;
        break;
    case q1:
        if (ch == '1')
            return q2;
        break;
    case q2:
        if (ch == '0')
            return q1;
        break;
    }
    return q0; // Default to q0 (invalid transition)
}
int main() {
    char input[20];
    enum states curr_state = q0;
    int i = 0;
    printf("Enter the string:");
    scanf("%s", input);
    while (input[i] != '\0') {
        curr_state = delta(curr_state, input[i]);
        i++;
    }
    if (curr_state == q2)
        printf("Accepted\n");
    else
        printf("Rejected\n");
    return 0;
}
```

Output 1:



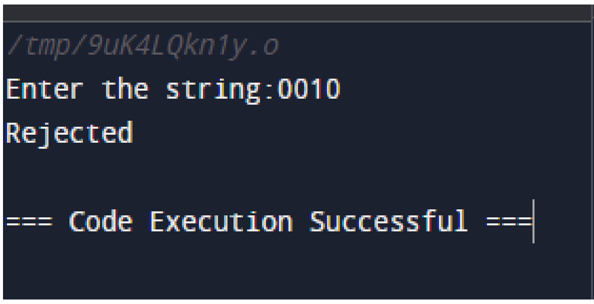
```
/tmp/zSJngTXsnI.o
Enter the string:01
Accepted

=== Code Execution Successful ===
```



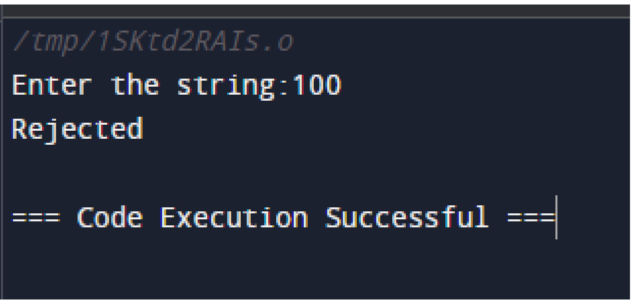
```
/tmp/sl5CO9Nilj.o
Enter the string:101
Accepted

=== Code Execution Successful ===
```



```
/tmp/9uK4LQkn1y.o
Enter the string:0010
Rejected

=== Code Execution Successful ===
```



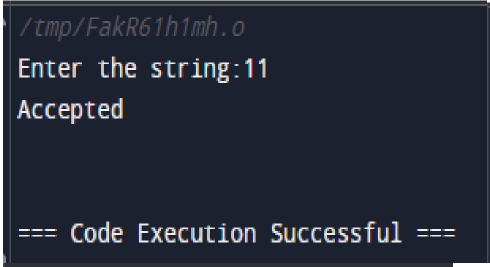
```
/tmp/1SKtd2RAIs.o
Enter the string:100
Rejected

=== Code Execution Successful ===
```

Discussion part :

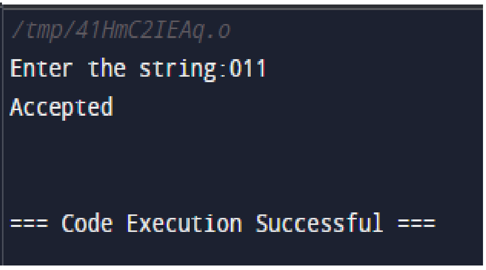
Input	Output	Discussion
01	Accepted	The NFA transitions from q0 to q1 on '0', then from q1 to q2 on '1'. Thus, the string is accepted.
101	Accepted	The NFA transitions from q0 to q0 on '1', then from q0 to q1 on '0', then from q1 to q2 on '1'. Thus, the string is accepted.
100	Rejected	The NFA transitions from q0 to q0 on '1', then from q0 to q0 on '0', but there is no transition from q0 to an accept state on the final '0'. Thus, the string is rejected.
0010	Rejected	The NFA transitions from q0 to q1 on '0', then fails to transition from q1 with input '0', leading to rejection.

Output 2:



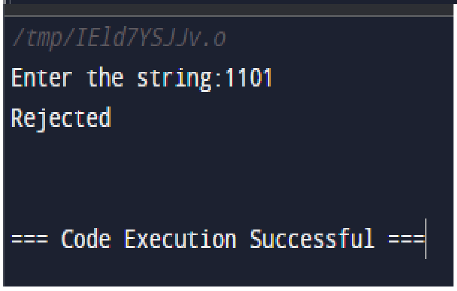
```
/tmp/FakR61h1mh.o
Enter the string:11
Accepted


=== Code Execution Successful ===
```



```
/tmp/41HmC2IEAq.o
Enter the string:011
Accepted


=== Code Execution Successful ===
```



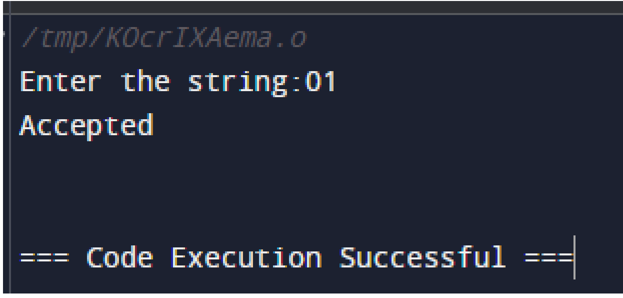
```
/tmp/IEld7YSJJv.o
Enter the string:1101
Rejected


=== Code Execution Successful ===
```

Discussion Part:

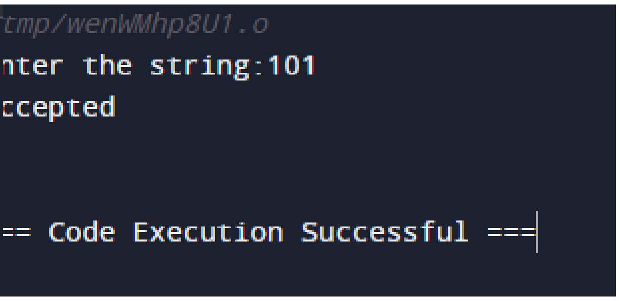
Input	Output	Description
11	Accepted	The NFA transitions from q0 to q1 on '1', then from q1 to q2 on '1'. Thus, the string is accepted.
011	Accepted	The NFA transitions from q0 to q1 on '0', then transition from q1 with input '1', leading to acceptance.
1101	Rejected	The NFA transitions from q0 to q1 on '1', then from q1 to q0 on '1', then from q0 to q1 on '0', then fails from q1 to q2 on '1'. Thus, the string is rejected.

Output 3:



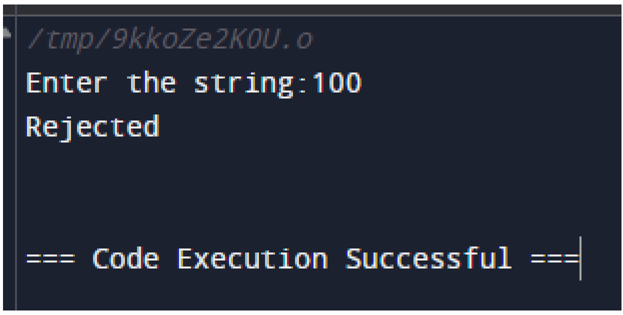
```
/tmp/KOcrIXAema.o
Enter the string:01
Accepted


=== Code Execution Successful ===
```



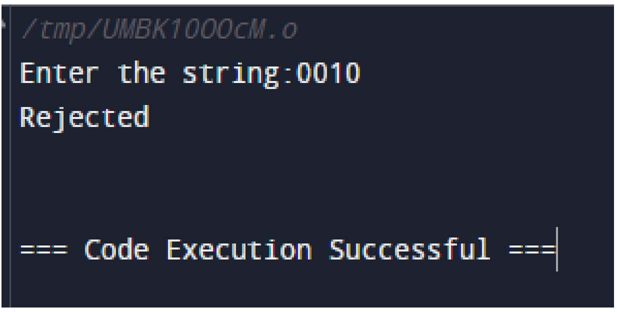
```
tmp/wenWMhp8U1.o
nter the string:101
ccepted


== Code Execution Successful ===
```



```
/tmp/9kkoZe2KOU.o
Enter the string:100
Rejected


=== Code Execution Successful ===
```



```
/tmp/UMBK10OOcM.o
Enter the string:0010
Rejected


=== Code Execution Successful ===
```

Discussion Part:

Input	**Output**	**Discussion**
01	Accepted	The NFA transitions from q0 to q1 on '0', then from q1 to q2 on '1'. Thus, the string is accepted.
101	Accepted	The NFA transitions from q0 to q0 on '1', then from q0 to q1 on '0', then from q1 to q2 on '1'. Thus, the string is accepted.
100	Rejected	The NFA transitions from q0 to q0 on '1', then from q0 to q0 on '0', but there is no transition from q0 to an accept state on the final '0'. Thus, the string is rejected.
0010	Rejected	The NFA transitions from q0 to q1 on '0', then fails to transition from q1 with input '0', leading to rejection.

4. CONCLUSION

The NFA for the specified language was successfully implemented using the C programming language. The NFA accurately determined whether input strings belonged to the language based on the given criteria. This experiment provided a comprehensive understanding of the NFA concept and its practical application.