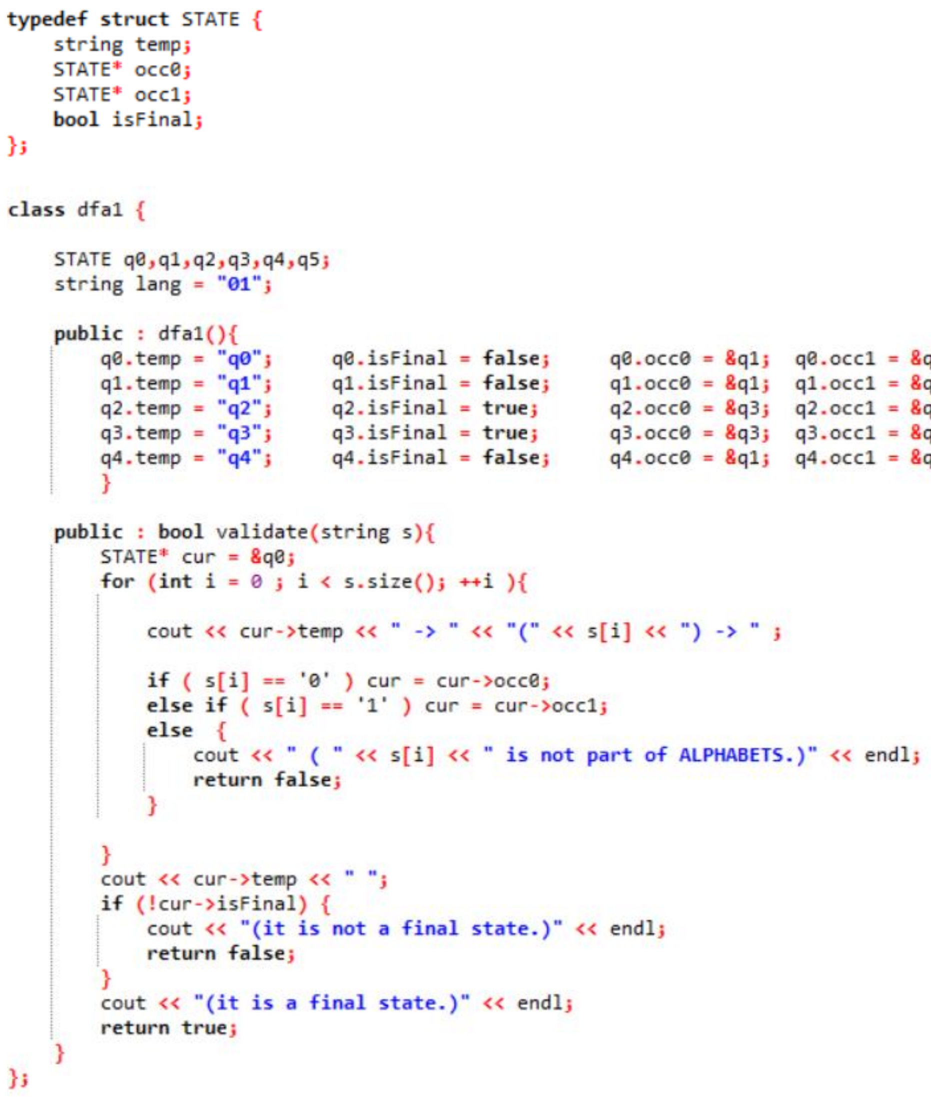
# TOA ASSIGNMENT #1

## SECTION F

| SR # | NAME | ROLL # |
|---|---|---|
| 1 | Mehdi Raza Rajani | K163904 |
| 2 | Tahir Raza Hemani | K163905 |
| 3 | Moazzam Maqsood | K163868 |

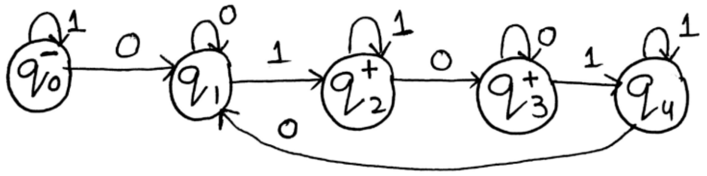
## DFA 1:

**DFA for the language {w : w contains 01 an odd number of times} over the alphabet { 0 , 1 }.**



```
typedef struct STATE {
    string temp;
    STATE* occ0;
    STATE* occ1;
    bool isFinal;
};

class dfa1 {

    STATE q0,q1,q2,q3,q4,q5;
    string lang = "01";

    public : dfa1(){
        q0.temp = "q0";     q0.isFinal = false;     q0.occ0 = &q1;  q0.occ1 = &q0;
        q1.temp = "q1";     q1.isFinal = false;     q1.occ0 = &q1;  q1.occ1 = &q2;
        q2.temp = "q2";     q2.isFinal = true;      q2.occ0 = &q3;  q2.occ1 = &q2;
        q3.temp = "q3";     q3.isFinal = true;      q3.occ0 = &q3;  q3.occ1 = &q4;
        q4.temp = "q4";     q4.isFinal = false;     q4.occ0 = &q1;  q4.occ1 = &q4;
        }

    public : bool validate(string s){
        STATE* cur = &q0;
        for (int i = 0 ; i < s.size(); ++i ){

            cout << cur->temp << " -> " << "(" << s[i] << ") -> " ;

            if ( s[i] == '0' ) cur = cur->occ0;
            else if ( s[i] == '1' ) cur = cur->occ1;
            else  {
                cout << " ( " << s[i] << " is not part of ALPHABETS.)" << endl;
                return false;
            }

        }
        cout << cur->temp << " ";
        if (!cur->isFinal) {
            cout << "(it is not a final state.)" << endl;
            return false;
        }
        cout << "(it is a final state.)" << endl;
        return true;
    }
};
```

**DFA:**



**TEST CASES:**

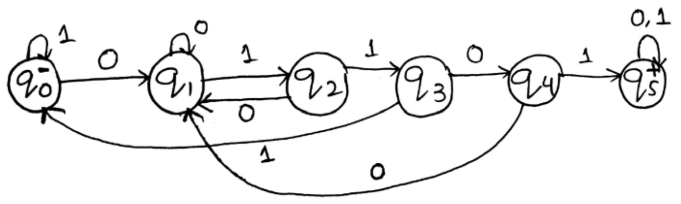
```
Enter you string you want to check for: 0101010
q0 -> (0) -> q1 -> (1) -> q2 -> (0) -> q3 -> (1) -> q4 -> (0) -> q1 -> (1) -> q2 -> (0) -> q3 (it is a final state.)
Yes!

Enter you string you want to check for: 1111010101111101
q0 -> (1) -> q0 -> (1) -> q0 -> (1) -> q0 -> (1) -> q0 -> (0) -> q1 -> (1) -> q2 -> (0) -> q3 -> (1) -> q4 -> (0) -> q1 -> (1) ->
q2 -> (1) -> q2 -> (1) -> q2 -> (1) -> q2 -> (1) -> q2 -> (0) -> q3 -> (1) -> q4 (it is not a final state.)
No!
```

## DFA 2:

**DFA for the language of words over the alphabet { 0 , 1 } that contain the substring 01101.**

```cpp
typedef struct STATE {
    string temp;
    STATE* occ0;
    STATE* occ1;
    bool isFinal;
};

class dfa2 {
    STATE q0,q1,q2,q3,q4,q5;
    string lang = "01";
    public : dfa2(){
        q0.temp = "q0";     q0.isFinal = false;     q0.occ0 = &q1;  q0.occ1 = &q0;
        q1.temp = "q1";     q1.isFinal = false;     q1.occ0 = &q1;  q1.occ1 = &q2;
        q2.temp = "q2";     q2.isFinal = false;     q2.occ0 = &q1;  q2.occ1 = &q3;
        q3.temp = "q3";     q3.isFinal = false;     q3.occ0 = &q4;  q3.occ1 = &q0;
        q4.temp = "q4";     q4.isFinal = false;     q4.occ0 = &q1;  q4.occ1 = &q5;
        q5.temp = "q5";     q5.isFinal = true;      q5.occ0 = &q5;  q5.occ1 = &q5;
    }
    public : bool validate(string s){
        STATE* cur = &q0;
        for (int i = 0 ; i < s.size(); ++i ){
            cout << cur->temp << " -> " << "(" << s[i] << ") -> " ;
            if ( s[i] == '0' ) cur = cur->occ0;
            else if ( s[i] == '1' ) cur = cur->occ1;
            else  {
                cout << " ( " << s[i] << " is not part of ALPHABETS.)" << endl;
                return false;
            }
        }
        cout << cur->temp << " ";
        if (!cur->isFinal) {
            cout << "(it is not a final state.)" << endl;
            return false;
        }
        cout << "(it is a final state.)" << endl;
        return true;
    }
};
```

**DFA:**



**TEST CASES:**

```
Enter you string you want to check for: 110110110
q0 -> (1) -> q0 -> (1) -> q0 -> (0) -> q1 -> (1) -> q2 -> (1) -> q3 -> (0) -> q4 -> (1) -> q5 -> (1) -> q5 -> (0) -> q5 (it is a final state.)
Yes!
```

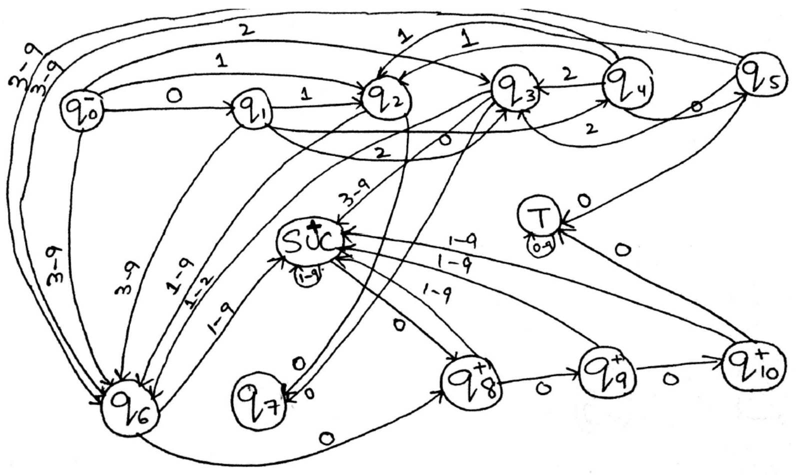
```
Enter you string you want to check for: 1101145
q0 -> (1) -> q0 -> (1) -> q0 -> (0) -> q1 -> (1) -> q2 -> (1) -> q3 -> (4) ->  ( 4 is not part of ALPHABETS.)
No!
```

## DFA 3:

**DFA for the language {w : w is an integer at least 23} over the alphabet { 0 , 1 , . . . , 9 }. Words in this language should not have max three leading 0s.**

```
typedef struct STATE2 {
    string temp;
    STATE2* occ0;
    STATE2* occ1;
    STATE2* occ2;
    STATE2* occ3p;
    bool isFinal;
};

class dfa3 {
    STATE2 q0,q1,q2,q3,q4,q5,q6,q7,q8,q9,q10,SUC,T;
    string lang = "0123456789";
    public : dfa3 (){
        q0.temp = "q0";     q0.isFinal = false;     q0.occ0 = &q1;      q0.occ1 = &q2;      q0.occ2 = &q3;      q0.occ3p = &q6;
        q1.temp = "q1";     q1.isFinal = false;     q1.occ0 = &q4;      q1.occ1 = &q2;      q1.occ2 = &q3;      q1.occ3p = &q6;
        q2.temp = "q2";     q2.isFinal = false;     q2.occ0 = &q7;      q2.occ1 = &q6;      q2.occ2 = &q6;      q2.occ3p = &q6;
        q3.temp = "q3";     q3.isFinal = false;     q3.occ0 = &q7;      q3.occ1 = &q6;      q3.occ2 = &q6;      q3.occ3p = &SUC;
        q4.temp = "q4";     q4.isFinal = false;     q4.occ0 = &q5;      q4.occ1 = &q2;      q4.occ2 = &q3;      q4.occ3p = &q6;
        q5.temp = "q5";     q5.isFinal = false;     q5.occ0 = &T;       q5.occ1 = &q2;      q5.occ2 = &q3;      q5.occ3p = &q6;
        q6.temp = "q6";     q6.isFinal = false;     q6.occ0 = &q8;      q6.occ1 = &SUC;     q6.occ2 = &SUC;     q6.occ3p = &SUC;
        q7.temp = "q7";     q7.isFinal = false;     q7.occ0 = &q9;      q7.occ1 = &SUC;     q7.occ2 = &SUC;     q7.occ3p = &SUC;
        q8.temp = "q8";     q8.isFinal = true;      q8.occ0 = &q9;      q8.occ1 = &SUC;     q8.occ2 = &SUC;     q8.occ3p = &SUC;
        q9.temp = "q9";     q9.isFinal = true;      q9.occ0 = &q10;     q9.occ1 = &SUC;     q9.occ2 = &SUC;     q9.occ3p = &SUC;
        q10.temp = "q10";       q10.isFinal = true;     q10.occ0 = &T;      q10.occ1 = &SUC;        q10.occ2 = &SUC;        q10.occ3p = &SUC;
        SUC.temp = "SUC";   SUC.isFinal = true;     SUC.occ0 = &q8;     SUC.occ1 = &SUC;    SUC.occ2 = &SUC;    SUC.occ3p = &SUC;
        T.temp = "T";       T.isFinal = false;      T.occ0 = &T;        T.occ1 = &T;        T.occ2 = &T;        T.occ3p = &T;
    }
    public : bool validate(string s){
        STATE2* cur = &q0;
        for (int i = 0 ; i < s.size(); ++i ){
            cout << cur->temp << " -> " << "(" << s[i] << ") -> " ;
            if ( s[i] == '0' ) cur = cur->occ0;
            else if ( s[i] == '1' ) cur = cur->occ1;
            else if ( s[i] == '2' ) cur = cur->occ2;
            else if ( s[i] >= '3' && s[i] <= '9' ) cur = cur->occ3p;
            else  {
                cout << " ( " << s[i] << " is not part of ALPHABETS.)" << endl;
                return false;
            }
        }
        cout << cur->temp << " ";
        if (!cur->isFinal) {
            cout << "(it is not a final state.)" << endl;
            return false;
        }
        cout << "(it is a final state.)" << endl;
        return true;
    }
};
```

**DFA:**

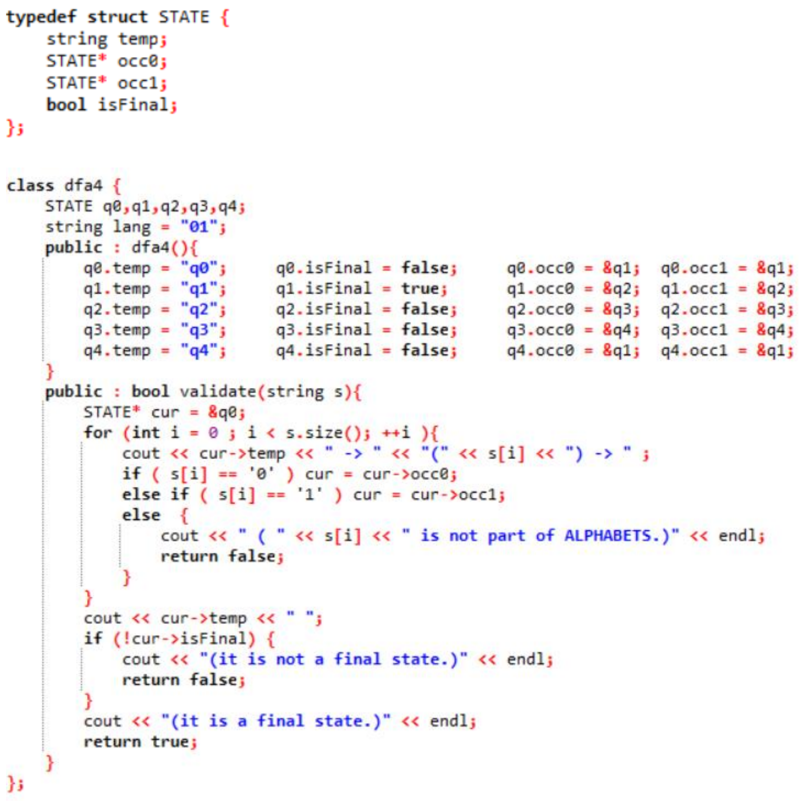


**TEST CASES:**

```
Enter you string you want to check for: 0000142
q0 -> (0) -> q1 -> (0) -> q4 -> (0) -> q5 -> (0) -> T -> (1) -> T -> (4) -> T -> (2) -> T (it is not a final state.)
No!
Enter you string you want to check for: 2354
q0 -> (2) -> q3 -> (3) -> SUC -> (5) -> SUC -> (4) -> SUC (it is a final state.)
Yes!
```

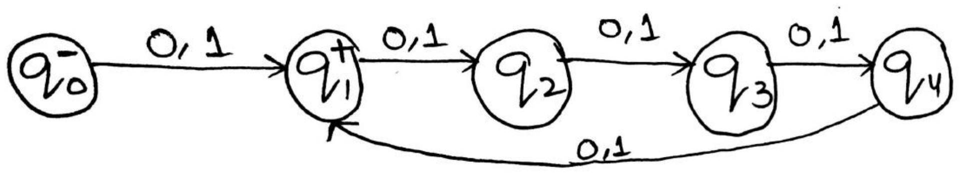
## DFA 4:

**S = { 0 , 1 } , L = {w e S|  |w| mod 4 = 1}.**



```
typedef struct STATE {
    string temp;
    STATE* occ0;
    STATE* occ1;
    bool isFinal;
};


class dfa4 {
    STATE q0,q1,q2,q3,q4;
    string lang = "01";
    public : dfa4(){
        q0.temp = "q0";     q0.isFinal = false;     q0.occ0 = &q1;  q0.occ1 = &q1;
        q1.temp = "q1";     q1.isFinal = true;      q1.occ0 = &q2;  q1.occ1 = &q2;
        q2.temp = "q2";     q2.isFinal = false;     q2.occ0 = &q3;  q2.occ1 = &q3;
        q3.temp = "q3";     q3.isFinal = false;     q3.occ0 = &q4;  q3.occ1 = &q4;
        q4.temp = "q4";     q4.isFinal = false;     q4.occ0 = &q1;  q4.occ1 = &q1;
    }
    public : bool validate(string s){
        STATE* cur = &q0;
        for (int i = 0 ; i < s.size(); ++i ){
            cout << cur->temp << " -> " << "(" << s[i] << ") -> " ;
            if ( s[i] == '0' ) cur = cur->occ0;
            else if ( s[i] == '1' ) cur = cur->occ1;
            else  {
                cout << " ( " << s[i] << " is not part of ALPHABETS.)" << endl;
                return false;
            }
        }
        cout << cur->temp << " ";
        if (!cur->isFinal) {
            cout << "(it is not a final state.)" << endl;
            return false;
        }
        cout << "(it is a final state.)" << endl;
        return true;
    }
};
```

**DFA:**

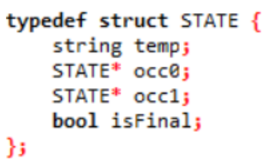


**TEST CASES:**

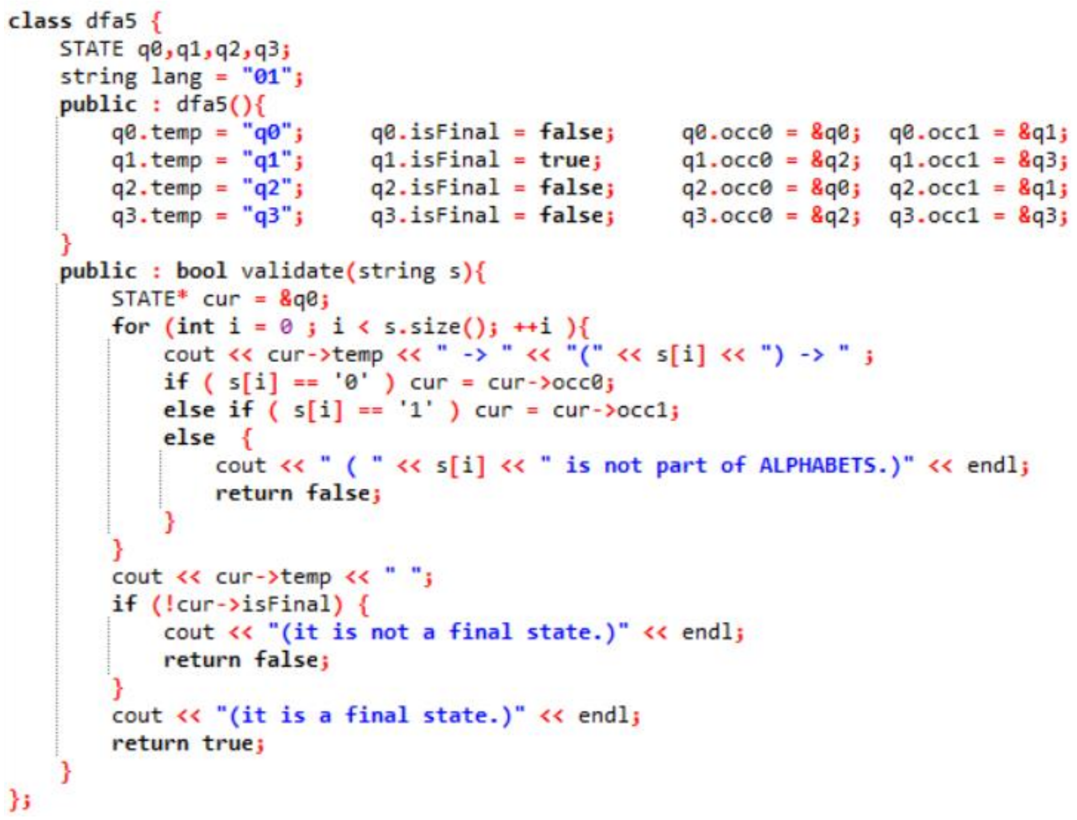
```
Enter you string you want to check for: 1010101
q0 -> (1) -> q1 -> (0) -> q2 -> (1) -> q3 -> (0) -> q4 -> (1) -> q1 -> (0) -> q2 -> (1) -> q3 (it is not a final state.)
No!

Enter you string you want to check for: 0101011110101
q0 -> (0) -> q1 -> (1) -> q2 -> (0) -> q3 -> (1) -> q4 -> (0) -> q1 -> (1) -> q2 -> (1) -> q3 -> (1) -> q4 -> (1) -> q1
-> (0) -> q2 -> (1) -> q3 -> (0) -> q4 -> (1) -> q1 (it is a final state.)
Yes!
```

## DFA 5:

**S = { 0 , 1 } , L = {w e S|  w mod 4 = 1}.**

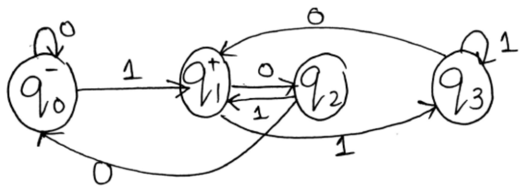


```
typedef struct STATE {
    string temp;
    STATE* occ0;
    STATE* occ1;
    bool isFinal;
};
```



```
class dfa5 {
    STATE q0,q1,q2,q3;
    string lang = "01";
    public : dfa5(){
        q0.temp = "q0";     q0.isFinal = false;     q0.occ0 = &q0;  q0.occ1 = &q1;
        q1.temp = "q1";     q1.isFinal = true;      q1.occ0 = &q2;  q1.occ1 = &q3;
        q2.temp = "q2";     q2.isFinal = false;     q2.occ0 = &q0;  q2.occ1 = &q1;
        q3.temp = "q3";     q3.isFinal = false;     q3.occ0 = &q2;  q3.occ1 = &q3;
    }
    public : bool validate(string s){
        STATE* cur = &q0;
        for (int i = 0 ; i < s.size(); ++i ){
            cout << cur->temp << " -> " << "(" << s[i] << ") -> " ;
            if ( s[i] == '0' ) cur = cur->occ0;
            else if ( s[i] == '1' ) cur = cur->occ1;
            else  {
                cout << " ( " << s[i] << " is not part of ALPHABETS.)" << endl;
                return false;
            }
        }
        cout << cur->temp << " ";
        if (!cur->isFinal) {
            cout << "(it is not a final state.)" << endl;
            return false;
        }
        cout << "(it is a final state.)" << endl;
        return true;
    }
};
```

## DFA:



## TEST CASES:

```
Enter you string you want to check for: 10011011
q0 -> (1) -> q1 -> (0) -> q2 -> (0) -> q0 -> (1) -> q1 -> (1) -> q3 -> (0) -> q2 -> (1) -> q1 -> (1) -> q3 (it is not a
final state.)
No!
Enter you string you want to check for: 00111001
q0 -> (0) -> q0 -> (0) -> q0 -> (1) -> q1 -> (1) -> q3 -> (1) -> q3 -> (0) -> q2 -> (0) -> q0 -> (1) -> q1 (it is
 a final state.)
Yes!
```