Name : Dasari Jayanth

Roll No : 2019BCS-016

Course :System Software Lab

Course Code : BCCS 3106 - 2021

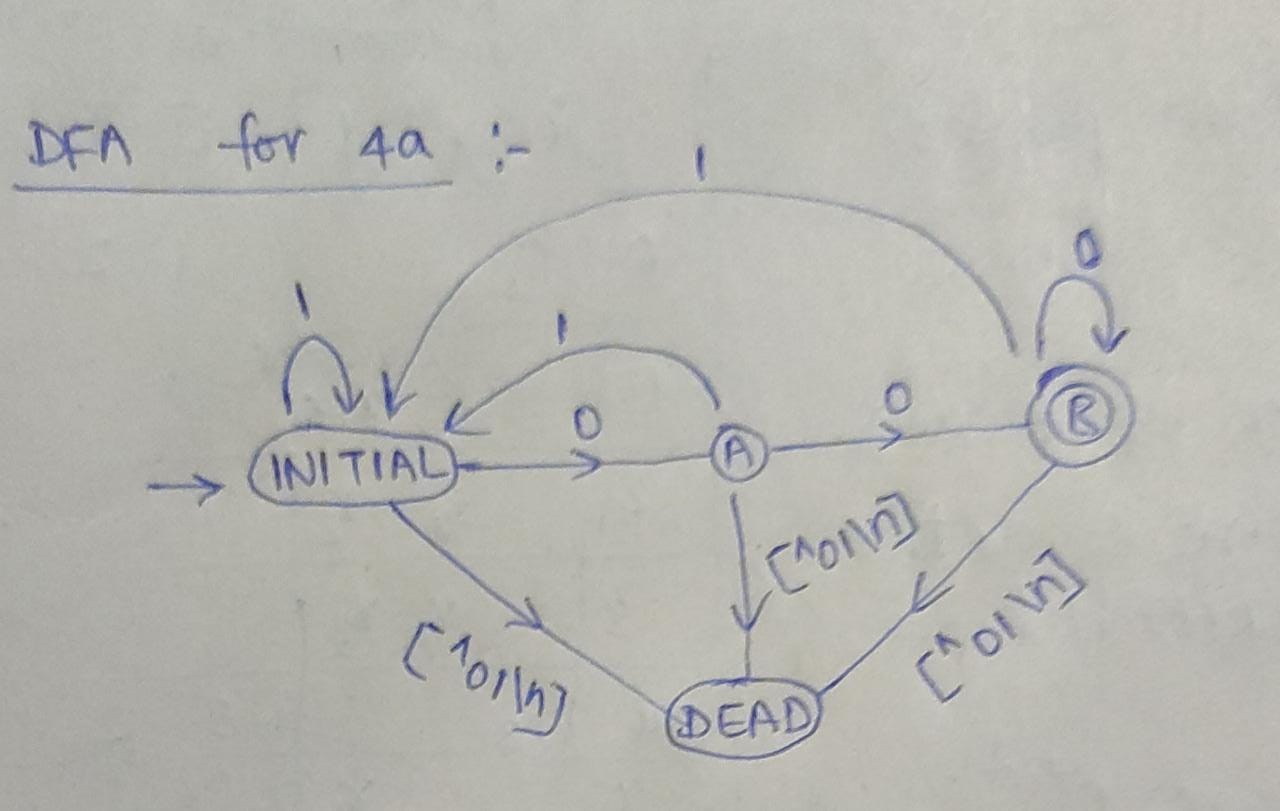
# **Assignment Number - 4a**

Date Of Submission: **03-10-2021**

**Aim:**

To write a program for the lex/flex scanner generator to construct a DFA which accepts the language in which all the strings end with “00” over inputs ‘0’ and ‘1’.

**Procedure:**



Above diagram represents the DFA which accepts the language ending with ‘00’ over ‘0’ and ‘1’.

At first we initialize the states ‘INITIAL’, ‘A’, ‘B’, ‘DEAD’. First we are at initial state, When the pointer pointing at the given user input string is ‘0’ then move to state ‘A’ else if it is ‘1’ then move to state ’INITIAL’ i.e to itself else move it to DEAD state as the inputs allowed are only ‘1’ and ‘0’. (After checking move the pointer to next char)

Now, When we are at state ‘A’ then if pointing at ‘0’ then move to state B which will be the final state because to reach this state string should contain 2 consecutive 0’s at least, if it is ‘1’ then move to INITIAL state else to dead state.

When we are at state B and still the string is not read completely then if it is ‘0’ then move to itself since the B is final state and last should contain 0 only and previously minimum 2 0’s condition satisfied, if it is ‘1’ then move to INITIAL state as it should end with 0, else to dead state. ACCEPT the string only if at the end of string it is at state ‘B’ else not accepted.

**Code:**

%{

%}

%s A B DEAD

%%

<INITIAL>0 BEGIN A;

<INITIAL>1 BEGIN INITIAL;

<INITIAL>[^01\n] BEGIN DEAD;

<INITIAL>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<A>0 BEGIN B;

<A>1 BEGIN INITIAL;

<A>[^01\n] BEGIN DEAD;

<A>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<B>0 BEGIN B;

<B>1 BEGIN INITIAL;

<B>[^01\n] BEGIN DEAD;

<B>\n BEGIN INITIAL; {printf("Accepted\n");}

<DEAD>[^\n] BEGIN DEAD;

<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}

%%

int yywrap() {}

int main()

{

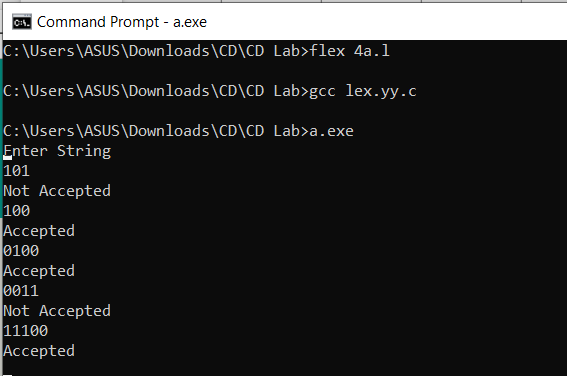
printf("Enter String\n");

yylex();

return 0;

}

**Input/Output -4a:**



**Inference:**

Input String: 101.

Output: Not Accepted.

Reasoning: Since ending with ‘1’ not ‘00’.

Input String: 100.

Output: Accepted.

Reasoning: Since ending with ‘00’.

**Hence**, we can conclude that the code written above is for DFA that accepts the given string if it ends with ‘00’ over inputs ‘1’, ‘0’.

# 

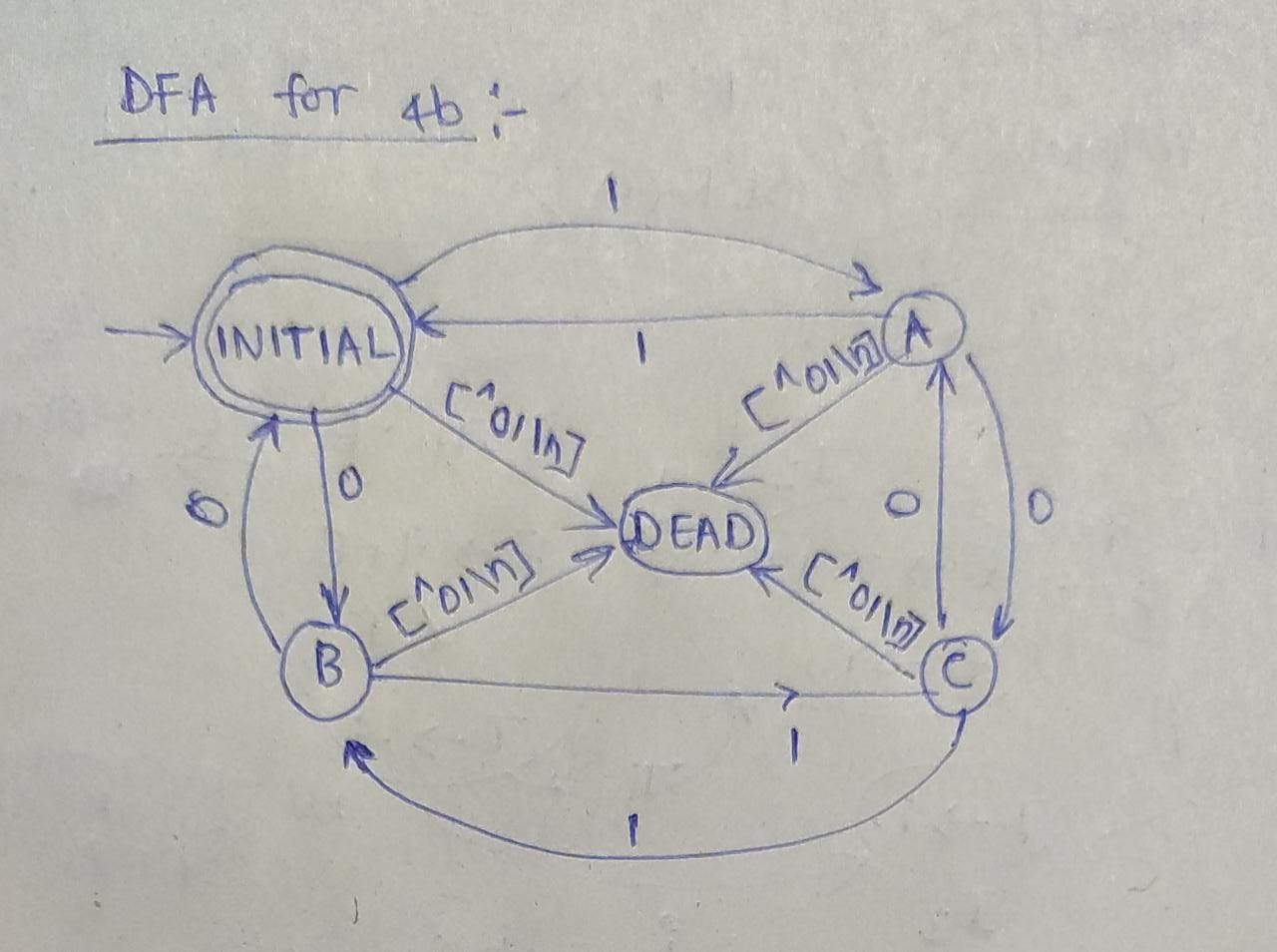
# **Assignment Number - 4b**

Date Of Submission: **03-10-2021**

**Aim:**

To write a program for the lex/flex scanner generator to construct a DFA which accepts the language in which all strings have an even number of 1’s and even number of 0's over inputs ‘0’ and ‘1’.

**Procedure:**



Above diagram is one of the possible DFA’s for the given aim.

Initialize states INITIAL, A, B, C, DEAD.

At first we are at the INITIAL state.

If the pointer pointing at string contains ‘0’ move to B else if ‘1’ move to A else to DEAD state.

At state ‘A’ if ‘1’ move to INITIAL else if ‘0’ move to ‘C’ else to DEAD state.

At state ‘B’ if ‘1’ move to C else if ‘0’ move to INITIAL else to DEAD state.

At state ‘C’ if ‘1’ move to B else if ‘0’ move to A else to DEAD state.

[^01\n] : means if the character is not ‘0’ or ‘1’ or ‘\n’ .

Here INITIAL state is the final state here b/c it comes to initial state only if the string upto that point has even number of 0’s and 1’s.

If at the end of the string we are at the final state ‘INITIAL’ then accept the string else not accepted.

**Code:**

%{

%}

%s A B C DEAD

%%

<INITIAL>1 BEGIN A;

<INITIAL>0 BEGIN B;

<INITIAL>[^01\n] BEGIN DEAD;

<INITIAL>\n BEGIN INITIAL; {printf("Accepted\n");}

<A>1 BEGIN INITIAL;

<A>0 BEGIN C;

<A>[^01\n] BEGIN DEAD;

<A>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<B>1 BEGIN C;

<B>0 BEGIN INITIAL;

<B>[^01\n] BEGIN DEAD;

<B>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<C>1 BEGIN B;

<C>0 BEGIN A;

<C>[^01\n] BEGIN DEAD;

<C>\n BEGIN INITIAL; {printf("Not Accepted\n");}

<DEAD>[^\n] BEGIN DEAD;

<DEAD>\n BEGIN INITIAL; {printf("Invalid\n");}

%%

int yywrap() {}

int main()

{

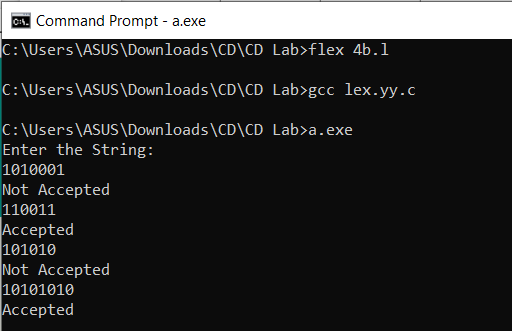
printf("Enter the String: \n");

yylex();

return 0;

}

**Input/Output -4b:**

****

**Inference:**

Input : 1010001.

Output : Not Accepted.

Reasoning: Since the given string doesn't contain even no. of 1’s. (1’s count= 3; 0’s count= 4).

Input: 110011.

Output: Accepted.

Reasoning: Since the given string contains even no. of 0’s and 1’s. (1’s count= 4; 0’s count= 2).

**Hence**, we can conclude that the code written above is for DFA that accepts the given string if it contains an even number of 0’s and 1’s over inputs ‘1’, ‘0’.