LAKSHMI NARAIN COLLEGE OF TECHNOLOGY & SCIENCE, BHOPAL

CS-501

Theory Of Computation

Vision of the Department

To be recognized for keeping innovation, research and excellence abreast of learning in the field of computer science & engineering to cater the global society.

Mission of the Department

M1: To provide an exceptional learning environment with academic excellence in the field of computer science and engineering.

M2: To facilitate the students for research and innovation in the field of software, hardware and computer applications and nurturing to cater the global society.

M3: To establish professional relationships with industrial and research organisations to enable the students to be updated of the recent technological advancements.

M4: To groom the learners for being the software professionals catering the needs of modern society with ethics, moral values and full of patriotism.

Program Educational Objectives (PEO's)

PEO1: The graduate will have the knowledge and skills of major domains of computer science and engineering in providing solution to real world problems most efficiently.

PEO2: The graduate will be able to create and use the modern tools and procedures followed in the software industry in the relevant domain.

PEO3: The graduate will be following the ethical practices of the software industry and contributing to the society as a responsible citizen.

PEO4: The graduate will have the innovative mindset of learning and implementing the latest developments and research outcomes in the computer hardware and software to keep pace with the fast changing socio economic world.

LAKSHMI NARAIN COLLEGE OF TECHNOLOGY & SCIENCE, BHOPAL

CS-501

Theory Of Computation

COURSE OUTCOMES

CO1: Compare deterministic and nondeterministic finite state machines

CO2: Sketch Finite State Machine and push down automata on various types of languages

CO3: Classify various types of languages based on recognizer and generator

CO4: Derive recursive and recursively enumerable languages

CO5: Design Turing machine as a capacitor with its capabilities_

LIST OF EXPERIMENTS

1. Design a Program for creating machine that accepts three consecutive one.
2. Design a Program for creating machine that accepts the string always ending with 101.
3. Design a Program for Mode 3 Machine.
4. Design a program for accepting decimal number divisible by 2.
5. Design a program for creating a machine which accepts string having equal no. of 1's and 0's.
6. Design a program for creating a machine which count number of 1's and 0's in a given string.
7. Design a Program to find 2's complement of a given binary number.
8. Design a Program which will increment the given binary number by 1.
9. Design a Program to convert NDFA to DFA.
10. Design a Program to create PDA machine that accept the well-formed parenthesis.

CS-501

Theory Of Computation

EXPERIMENT -1

Aim: Design a Program for creating machine that accepts three consecutive one.

Deterministic Finite Automaton (DFA):

In DFA, for each input symbol, one can determine the state to which the machine will move. Hence, it is called Deterministic Automaton. As it has a finite number of states, the machine is called Deterministic Finite Machine or Deterministic Finite Automaton.

Formal Definition of a DFA

A DFA can be represented by a 5-tuple (Q, ∑, δ, q_0, F) where −

- **Q** is a finite set of states.
- ∑ is a finite set of symbols called the alphabet.
- **δ** is the transition function where δ: Q × ∑ → Q
- **q_0** is the initial state from where any input is processed ($q_0 \in Q$).
- **F** is a set of final state/states of Q ($F \subseteq Q$).

Graphical Representation of a DFA

A DFA is represented by digraphs called **state diagram**.

- The vertices represent the states.
- The arcs labeled with an input alphabet show the transitions.
- The initial state is denoted by an empty single incoming arc.
- The final state is indicated by double circles.

Example

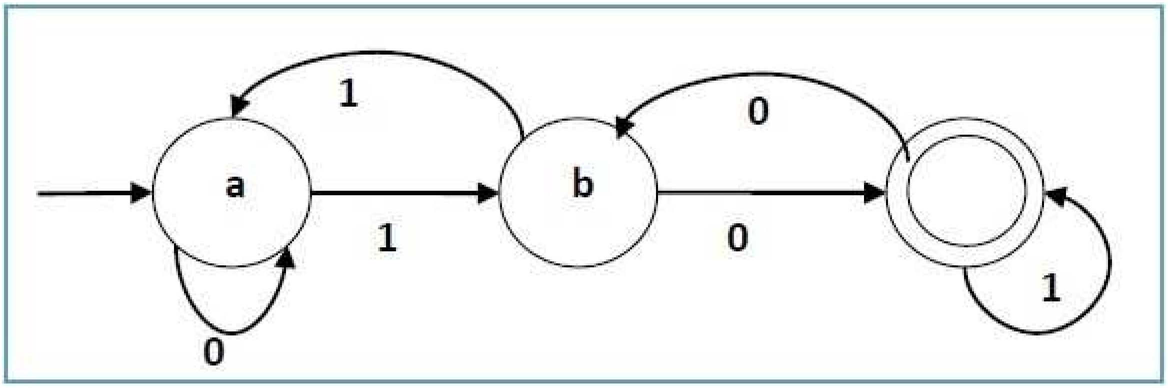
Let a deterministic finite automaton be →

- Q = {a, b, c},
- ∑ = {0, 1},
- q_0 = {a},
- F = {c}, and

Transition function δ as shown by the following table –

Present State	Next State for Input 0	Next State for Input 1
a	a	b
b	c	a
c	b	c

Its graphical representation would be as follows –



CS-501

Theory Of Computation

- **Design a program for creating machine that accept the three consecutive one.**

LAKSHMI NARAIN COLLEGE OF TECHNOLOGY & SCIENCE, BHOPAL

CS-501

Theory Of Computation

QUESTIONS

Q.1 Design a DFA with accept all string contain even number of 1s over alphabet 1.

……………………………………………………………………………………………………

…………………………………………………………………………………………………….

………………………………………………………………………………………………………

………………………………………………………………………………………………………

………………………………………………………………………………………………………

………………………………………………………………………………………………………

………………………………………………………………………………………………………
.

Q. 2 Show that L={0^i1^i/ i greater than equal to 1} is not regular.

……………………………………………………………………………………………………..

………………………………………………………………………………………………………

………………………………………………………………………………………………………

………………………………………………………………………………………………………

………………………………………………………………………………………………………

……………………………………………………………………………………………………..
…….

………………………………………………………………………………………………………

……………………………………………………………………………………………

………………………………………………………………………………………………………

……………………………………………………………………………………………………..

………………………………………………………………………………………………………

Q 3.Show that L={a^p/ p is a prime } is not regular.

…………………………………………………………………………………………………

………………………………………………………………………………………………..
…….

…………………………………………………………………………………………………

……………………………………………………………………………………………….

……..……………………………………………………………………………………………….

LAKSHMI NARAIN COLLEGE OF TECHNOLOGY & SCIENCE, BHOPAL

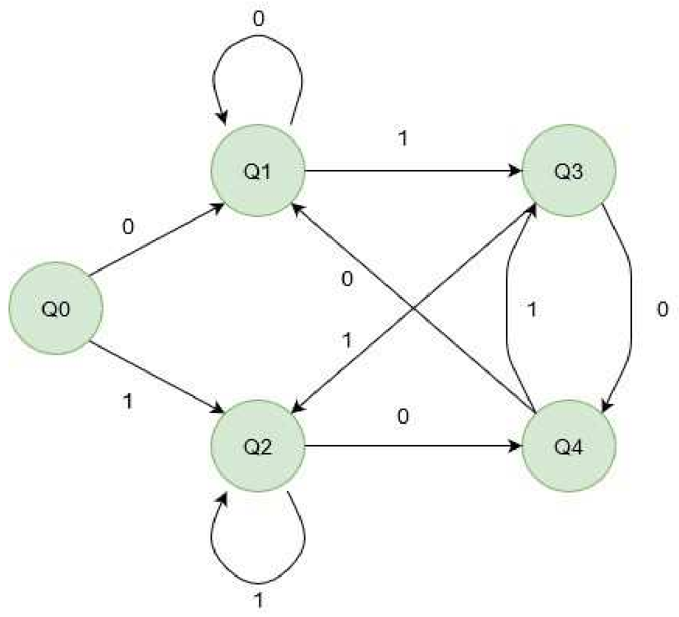
CS-501

Theory Of Computation

EXPERIMENT -2

Aim: Design a Program for creating machine that accepts the string always ending with 101.

DFA or Deterministic Finite Automata is a finite state machine which accepts a string(under some specific condition) if it reaches a final state, otherwise rejects it.
Problem: Given a string of ‘0’s and ‘1’s character by character, check for the last two characters to be “01” or “10” else reject the string. Also print the state diagram irrespective of acceptance or rejection. Since in DFA, there is no concept of memory, therefore we can only check for one character at a time, beginning with the 0th character. The input set for this problem is {0, 1}. For each character in the input set, each state of DFA redirects to another valid state.
DFA Machine: For the above problem statement, we must first build a DFA machine. DFA machine is similar to a flowchart with various states and transitions. DFA machine corresponding to the above problem is shown below, Q3 and Q4 are the final states:



Examples:
Input: 010101

Output:
```
State transitions are q0->q1->q3->q4
->q3->q4->q3->YES
```

Explanation : 010101 ends with "**01**".

Algorithm:

1. Define the minimum number of states required to make the state diagram. Use functions to define various states.
2. List all the valid transitions. Each state must have a transition for every valid symbol.
3. Define the final states by applying the base condition.
4. Define all the state transitions using state function calls.
5. Define a returning condition for the end of the string.

For the given DFA Machine:

1. Q0, Q1, Q2, Q3, Q4 are defined as the number of states.
2. 0 and 1 are valid symbols. Each state has transitions for 0 and 1.
3. Q3 and Q4 are defined as the final states.
4. Suppose at state Q0, if 0 comes, the function call is made to Q1. So, if 1 comes, the function call is made to Q2.
5. If the program reaches the end of the string, the output is made according to the state, the program is at.

Program to DFA that accepts string ending with 01 or 10.

```
#include <bits/stdc++.h>
using namespace std;

// Various states of DFA machine are defined
// using functions.
void q1(string, int);
void q2(string, int);
void q3(string, int);
void q4(string, int);

// End position is checked using the string
// length value.
// q0 is the starting state.
// q1 and q2 are intermediate states.
// q3 and q4 are final states.
void q1(string s, int i)
{
    cout << "q1->";

    if (i == s.length()) {
        cout << "NO \n";
        return;
    }

    // state transitions
    // 0 takes to q1, 1 takes to q3
    if (s[i] == '0')
        q1(s, i + 1);
    else
        q3(s, i + 1);
}

void q2(string s, int i)
{
    cout << "q2->";
    if (i == s.length()) {
        cout << "NO \n";
        return;
    }

    // state transitions
    // 0 takes to q4, 1 takes to q2
    if (s[i] == '0')
        q4(s, i + 1);
    else
        q2(s, i + 1);
}

void q3(string s, int i)
{
    cout << "q3->";
    if (i == s.length()) {
        cout << "YES \n";
        return;
    }

    // state transitions
    // 0 takes to q4, 1 takes to q2
    if (s[i] == '0')
        q4(s, i + 1);
    else
        q2(s, i + 1);
```

```
}

void q4(string s, int i)
{
   cout << "q4->";
   if (i == s.length()) {
      cout << "YES \n";
      return;
   }

   // state transitions
   // 0 takes to q1, 1 takes to q3
   if (s[i] == '0')
      q1(s, i + 1);
   else
      q3(s, i + 1);
}

void q0(string s, int i)
{
   cout << "q0->";
   if (i == s.length()) {
      cout << "NO \n";
      return;
   }

   // state transitions
   // 0 takes to q1, 1 takes to q2
   if (s[i] == '0')
      q1(s, i + 1);
   else
      q2(s, i + 1);
}

// Driver Code
int main()
{
   string s = "010101";
   // all state transitions are printed.
   // if string is accpetable, YES is printed.
   // else NO is printed
   cout << "State transitions are ";
   q0(s, 0);
}
```

Output:

```
State transitions are q0->q1->q3->q4->q3->q4->q3->YES
```

VIVA QUESTIONS

1. **Write the difference between NFA and DFA?**

   ……
   ……
   ………..

2. **What is the full form of NFA?**

   ……
   ……
   ………..

3. **Let ∑= {a, b, …. z} and A = {Hello, World}, B= {Input, Output}, then (A*∩B) U (B*∩A) can be represented as:?**

   ……
   ……
   ………..

4. **Number of states require to accept string ends with 10.?**

   ……
   ……
   ……

5. **δ*(q,ya) is equivalent to?**

   ……
   ……
   ………..

   ……

EXPERIMENT -3

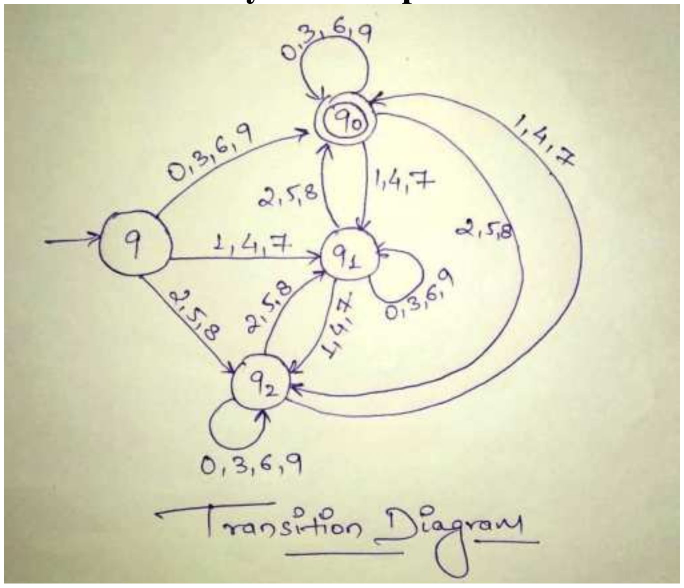
Aim: Design a Program for Mode 3 Machine.

State	Input		
-	**0, 3, 6, 9**	**1, 4, 7**	**2, 5, 8**
q	q0	q1	q2
q0	q0	q1	q2
q1	q1	q2	q0
q2	q2	q0	q1

As per the AIM, set of valid strings are represented by set A:
A = {0, 3, 6, 9, 03, 06, 09, 12, 012, ..}
means any decimal number string that when divided by three gives remainder zero. Let M be the machine for above AIM, hence it can be define as M(Q, Σ, δ, q0, F) where
Q: set of states: {q, q0, q1, q2}
Σ: set of input symbols: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
q0: initial state (q)
F: set of Final states: {q0}
δ: Transition Function: (Transition state diagram is shown in Figure 1.)

CS-501

Theory Of Computation



Code:

```
#include <iostream.h>
#include <conio.h>
#include <stdio.h>
void main()
{
        char Input[100];
        clrscr();
        cout<<"Enter a string to validate (input string should be decimal number (i.e
constructed from 0,1,2,3,4,5,6,7,8,9 digits)\n";
        gets(Input);
        int i=-1;
        q:
                i++;
                if(Input[i]=='0'|| Input[i]=='3'|| Input[i]=='6'|| Input[i]=='9')
                {
                        goto q0;
                }
                else if(Input[i]=='1'|| Input[i]=='4'|| Input[i]=='7')
                {
                        goto q1;
                }
                else if(Input[i]=='2'|| Input[i]=='5'|| Input[i]=='8')
                {
                        goto q2;
                }
                else if(Input[i]=='\0')
                {
                        goto Invalid;
                }
                else
                {
                        goto Wrong;
                }
        q0:
                i++;
                if(Input[i]=='0'|| Input[i]=='3'|| Input[i]=='6'|| Input[i]=='9')
                {
                        goto q0;
                }
                else if(Input[i]=='1'|| Input[i]=='4'|| Input[i]=='7')
                {
                        goto q1;
                }
                else if(Input[i]=='2'|| Input[i]=='5'|| Input[i]=='8')
                {
                        goto q2;
                }
                else if(Input[i]=='\0')
                {
                        goto Valid;
                }
                else
                {
                        goto Wrong;
                }
        q1:
                i++;
                if(Input[i]=='0'|| Input[i]=='3'|| Input[i]=='6'|| Input[i]=='9')
                {
                        goto q1;
                }
                else if(Input[i]=='1'|| Input[i]=='4'|| Input[i]=='7')
                {
                        goto q2;
                }
                else if(Input[i]=='2'|| Input[i]=='5'|| Input[i]=='8')
                {
                        goto q0;
                }
                else if(Input[i]=='\0')
```

```
            {
                    goto Invalid;
            }
            else
            {
                    goto Wrong;
            }
    q2:
            i++;
            if(Input[i]=='0'|| Input[i]=='3'|| Input[i]=='6'|| Input[i]=='9')
            {
                    goto q2;
            }
            else if(Input[i]=='1'|| Input[i]=='4'|| Input[i]=='7')
            {
                    goto q0;
            }
            else if(Input[i]=='2'|| Input[i]=='5'|| Input[i]=='8')
            {
                    goto q1;
            }
            else if(Input[i]=='\0')
            {
                    goto Invalid;
            }
            else
            {
                    goto Wrong;
            }
    Valid:
            cout<<"\n Output: Valid String";
            goto exit;
    Invalid:
            cout<<"\n Output: Invalid String";
            goto exit;
    Wrong:
            cout<<"\n Please enter valid decimal number string";
    exit:
            getch();
}
```

CS-501

Theory Of Computation

VIVA QUESTIONS

1. **Number of final state requires to accept Φ in minimal finite automata?**

………
………
……….

2. **Subset Construction method refers to:**

………
………
……….

3. **It is less complex to prove the closure properties over regular languages using?**

………
………
………

4. **Predict the number of transitions required to automate the following language using only 3 states:**

   L = {w | w ends with 00}?

   ………
   ………
   ………

5. **The total number of states to build the given language using DFA:**

   L = {w | w has exactly 2 a's and at least 2 b's}

   ………
   ………
   ………

EXPERIMENT -3

Aim: Design a program for accepting decimal number divisible by 2.

Code:

```
def stateq0(n):
   #if length found 0
   #print not accepted
   if (len(n)==0):
      print("string accepted")
   else:

      #if at index 0
      #'0' found call
      #function stateq0
```

```
        if(n[0]=='0'):
            stateq0(n[1:])

        #else if '1' found
        #call function q1.
        elif (n[0]=='1'):
            stateq1(n[1:])

def stateq1(n):
    #if length found 0
    #print not accepted
    if (len(n)==0):
        print("string not accepted")
    else:

        #if at index 0
        #'0' found call
        #function stateq0
        if(n[0]=='0'):
            stateq0(n[1:])

        #else if '1' found
        #call function q1.
        elif (n[0]=='1'):
            stateq1(n[1:])

#take number from user
n=int(input())
#converting number to binary
n = bin(n).replace("0b", "")

#to check the input
stateq0(n)
```

VIVA QUESTIONS

1. **What do you mean by Moore machine?**

………
………
……

2. **How Moore is differ from Mealy machine?**

……

……

3. **In Moore machine, output is produced over the change of?**

……
…….

……
…….

4. **Explain Mealy machine?**

……
…….

……
…….

EXPERIMENT -5

Aim: Design a program for creating a machine which accepts string having equal no. of 1's and 0's.

Source Code:

```
#include <bits/stdc++.h>
using namespace std;

// Function to find the count of substrings with equal no.
// of consecutive 0's and 1's
int countSubstring(string& S, int& n)
{
  // To store the total count of substrings
  int ans = 0;
  int i = 0;

  // Traversing the string
  while (i < n) {

  // Count of consecutive 0's & 1's
      int cnt0 = 0, cnt1 = 0;

  // Counting subarrays of type "01"
  if (S[i] == '0') {

  // Count the consecutive 0's
  while (i < n && S[i] == '0') {
  cnt0++;
  i++;
              }

  // If consecutive 0's ends then check for
  // consecutive 1's
  int j = i;

  // Counting consecutive 1's
  while (j < n && S[j] == '1') {
  cnt1++;
  j++;
              }
      }

      // Counting subarrays of type "10"
      else {

      // Count consecutive 1's
      while (i < n && S[i] == '1') {
      cnt1++;
      i++;
      }

      // If consecutive 1's ends then check for
      // consecutive 0's
      int j = i;

      // Count consecutive 0's
      while (j < n && S[j] == '0') {
      cnt0++;
      j++;
```

```
            }
        }

    // Update the total count of substrings with minimum
    // of (cnt0, cnt1)
    ans += min(cnt0, cnt1);
    }

    // Return answer
    return ans;
}

// Driver code
int main()
{
    string S = "0001110010";
    int n = S.length();

    // Function to print the count of substrings
    cout << countSubstring(S, n);
    return 0;
}
```

VIVA QUESTIONS

1. **Give an example of DFA?**

   ……
   ……
   ……

2. **Design a NFA by taking a string a string ?**

   ……

   ……

3. **What is language acceptor?**

   ……
   ……

………
……

4. **Explain automata with epsilon transition?**

………
………

………
……

5. **What do you mean by regular language?**

………
……….

………
………

EXPERIMENT NO. 6

Aim - Design a program for creating a machine which count number of 1's and 0's in a given string.

Source Code:

```
function count(str) {

 var countForZero = 0;

  var countForOne = 0;

  for (var i = 0, length = str.length; i < length; i++) {

    if (str[i] === '0') {
      countForZero++;

    }

    else {

      countForOne++;

    }

  }

      return {

    'zero': countForZero,

    'One': countForOne

  };

}
```

VIVA QUESTIONS

1. **What do you mean by pumping lemma?**

   ……
   ……
   ……

2. **What do you mean by regular expression?**

   ………… …….

   ………

3. **Differentiate between Moore and Mealy Machine?**

………
……

………
…….

4. **An automaton that presents output based on previous state or current input:**

………
…….

………
…….

Experiment-7

Aim: Design a Program to find 2's complement of a given binary number.

Discussion:

As per the AIM, Input can be any binary string and we have to design a turing machine that can calculate its two's complement.
For example (as shown in Figure 1) if the input binary string is 1011010000 then its two's complement will be 0100110000. For constructing a turing machine if we look in the logic then if we read the input string from right to left then the output string will remain exactly the same until the first 1 is found and as the first 1 encounter, complement of rest string appear in the output.

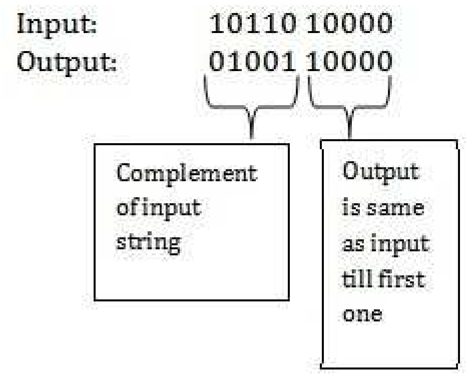

Input: 10110 10000
Output: 01001 10000
Complement of input string
Output is same as input till first one

Figure 1: Input output relationship of 2's complement of binary string while reading right to left.

Let M be the Turing Machine for above AIM, hence it can be define as M(Q, Σ, Γ, δ, q0, B, F) where

Q: set of states: {q0, q1, q2}

Σ: set of input symbols: {0, 1}

Γ: Set of Tape symbols: {0, 1, B}

q0: initial state (q0)

B: Blank Symbol (B)

F: set of Final states: { } turing machine as enumerator.]

δ: Transition Function:

State	Input		
-	0	1	B

State	Input		
q0	(q0, 0, R)	(q0, 1, R)	(q1, B, L)
q1	(q1, 0, L)	(q2, 1, L)	
q2	(q2, 1, L)	(q2, 0, L)	(q3, B, R)
q3			

Rules for implementing turing machine for a given language

Initial Setup: Load the input string in input buffer and consider the machine in at initial state q0.

Rules:

1. Move the input head at right direction, till it reaches to B. If B encounter then change state from q0 to q1.
2. Now, start reading input symbol right to left one by one.
3. If the input symbol is 0, move the input head left and do the same for all 0 till 1 encounter.
4. If the input symbol is 1, move the input head left and change its state from q1 to q2.
5. Now, if the input symbol is 0, make it 1 and if input symbol is 1 make it 0 till reach to the starting point.

Code:

```
#include <iostream.h>

#include <conio.h>

#include <stdio.h>

void main()

{

        char Input[100];

        clrscr();

        cout<<"Enter input binary string\n";

        gets(Input);

        int i=0;

        q0:

                if(Input[i]=='0'|| Input[i]=='1')

                {

                        i++;

                        goto q0;

                }

                else if(Input[i]=='\0')

                {

                        i--;

                        goto q1;

                }
        else
        {
                goto Invalid;
        }
q1:
        if(Input[i]=='0')
        {
                i--;
                goto q1;
        }
        else if(Input[i]=='1')
        {
                i--;
                goto q2;
        }
        else
        {
                goto Invalid;
        }
q2:
        if(Input[i]=='0')
        {
            Input[i]='1';
            i--;
            goto q2;
        }
        else if(Input[i]=='1')
        {
            Input[i]='0';
            i--;
            goto q2;
        }
        else if(i== -1)
        {
            goto q3;
        }
        else
        {
            goto Invalid;
        }
    q3:
        cout<<"\nOutput: Two's complement is";
        puts(Input);

        goto exit;

    Invalid:

        cout<<"\n You have entered some invalid string.";

        goto exit;

    exit:

        getch();

}
```

VIVA QUESTIONS

What is Turing Machine?

……
……
………

1. **How many tuples are used to define turing machine? Write the tuples of turing machine?**

   ………… ……….

   ……

2. **Give an example of turing machine?**

   ……
   …….

   ……
   …….

3. **How Turing machine is different from finite automata?**

   ……
   ………..

   ……
   …….

EXPERIMENT NO. 8

CS-501

Theory Of Computation

Aim: Design a Program which will increment the given binary number by 1.

We have to design a Turing Machine for incrementing the Binary Number by 1.

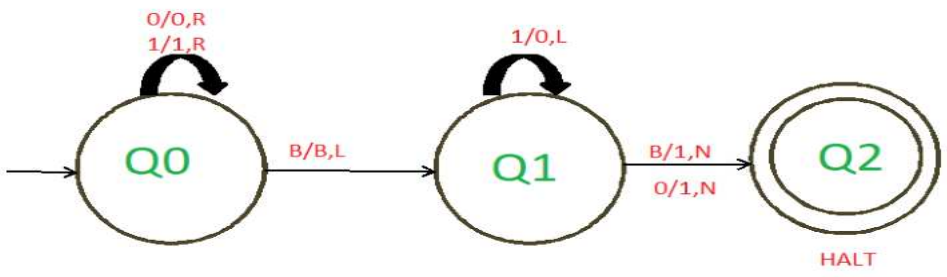
From the above three examples, we can get two conditions –

- **When the Rightmost digit is 0:**
  Here we can see that when we add something to a binary number having 0 as its rightmost digit, then the rightmost digit of the Binary number changes i.e. if the rightmost digit is 0 it will change to 1 and vice versa while all other digits remain the same and our machine will halt when we get Blank (B).
- **When the Rightmost digit is 1:**
  Here we can see that when we add something to a binary number having 1 as its rightmost digit, then all the 1's changes to 0's until we get a 0, and the 0 we get will change to 1 while all other digits after that will remain same and our machine will halt when we get Blank (B). Suppose we don't have any 0 in a string for example 1111 then we move to the left until we get a Blank (B) changing all the 1's to 0's and change this Blank (B) to 1, and our machine halts.

Approach:

1. We have to scan the element from right to left. At first, our pointer is at the leftmost side. Therefore we have to shift the pointer to the rightmost side.
2. To shift the pointer to the rightmost side we normally skip all 0's and 1's until we get a Blank (B).
3. After this step, we can now move the pointer from left to right.
4. If we get the first digit 1 then we change all the 1's to 0's until we get a 0 and change this 0 to 1. After this, all the digit remains the same, and our machine halts at Blank (B).
5. If we get the first digit 1 then a condition arises that we don't get any 0 for example 1111, then we move to the left until we get a Blank(B) changing all the 1's to 0's and change this Blank(B) to 1, and our machine halts.

6. If we get the first digit as 0 then we have to change 0 as 1 and after that, all the digit remains the same and our machine will halt at Blank (B).



VIVA QUESTIONS

LAKSHMI NARAIN COLLEGE OF TECHNOLOGY & SCIENCE, BHOPAL

CS-501

Theory Of Computation

1. **Write a Program for creating machine that accepts the string always ending with 101.?**

……
……
……

……
……
……

……
……
……

……
……
……

……
……
……

……
……
……

……
……
……

……
……
……

……
……
……

2. **Write a Program for creating machine that accepts three consecutive one.?**

………… ……………………………………………………………………………………………………

……

………… ………

………

………… ………

………

………… ………

………

………… ………

………

………… ………

………

………… ………

………

………… ………

………

………… ………

………

………… ………

EXPERIMENT NO. 9

Aim: Design a Program to convert NDFA to DFA.

An NFA can have zero, one or more than one move from a given state on a given input symbol. An NFA can also have NULL moves (moves without input symbol). On the other hand, DFA has one and only one move from a given state on a given input symbol.

Steps for converting NFA to DFA:

Step 1: Convert the given NFA to its equivalent transition table

To convert the NFA to its equivalent transition table, we need to list all the states, input symbols, and the transition rules. The transition rules are represented in the form of a matrix, where the rows represent the current state, the columns represent the input symbol, and the cells represent the next state.

Step 2: Create the DFA's start state

The DFA's start state is the set of all possible starting states in the NFA. This set is called the "epsilon closure" of the NFA's start state. The epsilon closure is the set of all states that can be reached from the start state by following epsilon (λ) transitions.

Step 3: Create the DFA's transition table

The DFA's transition table is similar to the NFA's transition table, but instead of individual states, the rows and columns represent sets of states. For each input symbol, the corresponding cell in the transition table contains the epsilon closure of the set of states obtained by following the transition rules in the NFA's transition table.

Step 4: Create the DFA's final states

The DFA's final states are the sets of states that contain at least one final state from the NFA.

Step 5: Simplify the DFA

The DFA obtained in the previous steps may contain unnecessary states and transitions. To simplify the DFA, we can use the following techniques:

Remove unreachable states: States that cannot be reached from the start state can be removed from the DFA.

Remove dead states: States that cannot lead to a final state can be removed from the DFA.

Merge equivalent states: States that have the same transition rules for all input symbols can be merged into a single state.

Step 6: Repeat steps 3-5 until no further simplification is possible

After simplifying the DFA, we repeat steps 3-5 until no further simplification is possible. The final DFA obtained is the minimized DFA equivalent to the given NFA**.**

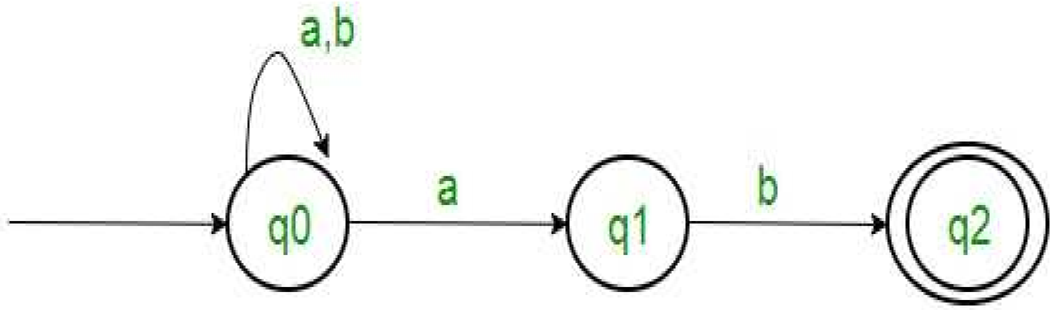


Figure 1

VIVA QUESTIONS

3. What is Halt in turing machine?

……
……
…….

4. What do you mean by Push down Automata?

………… ……….

……

5. What is regular expression?

……
………..

……
………..

6. Design a DFA for string 01010101.

……
……

……
………

7. Explain Push down automata with example.

……
………..

………

EXPERIMENT NO. 10

Aim: Design a Program to create PDA machine that accept the well-formed parenthesis.

Pushdown Automata (PDA) are the finite automata (FAs), but with the ability to push and pop symbols to/from a stack.

PDA accepts strings if there is a legal path from start state to acceptance state for input. Otherwise, the string is rejected.

A PDA can be represented by a 7-tuple

(Q, ∑, Γ, q0, ha, Δ,δ)

Where

The PDA is to finite subsets of Q X (Γ ∪ {Δ})*.

Parentheses are balanced if
While reading string, number of opening parentheses >= number of closing parentheses.
When string is read, number of opening parentheses = number of closing parentheses.
Examples
(())() − Balanced
((()() − Not balanced
)()(() − Not balanced
The context free grammar (CFG) is as follows −

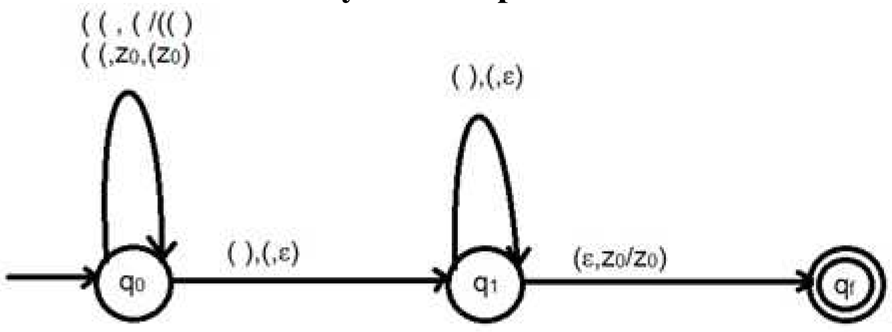
S -> (S) | SS | ε

δ(q0,(,z0)=(q0,(z0)

δ(q0,(,()=(q0,(()

δ(q1,),()=(q1,ε)

δ(q0,ε,z0)=(qf,ε)



Source Code:

```
// C++ program to check for balanced brackets.

#include <bits/stdc++.h>
using namespace std;

// Function to check if brackets are balanced
bool areBracketsBalanced(string expr)
{
  // Declare a stack to hold the previous brackets.
  stack<char> temp;
  for (int i = 0; i < expr.length(); i++) {
      if (temp.empty()) {

  // If the stack is empty
  // just push the current bracket
  temp.push(expr[i]);
      }
  else if ((temp.top() == '(' && expr[i] == ')')
  || (temp.top() == '{' && expr[i] == '}')
  || (temp.top() == '[' && expr[i] == ']')) {

  // If we found any complete pair of bracket
  // then pop
  temp.pop();
  }
  else {
  temp.push(expr[i]);
      }
  }
  if (temp.empty()) {
```

```
  // If stack is empty return true
  return true;
  }
  return false;
}

// Driver code
int main()
{
  string expr = "{()}[]";

  // Function call
  if (areBracketsBalanced(expr))
      cout << "Balanced";
  else
      cout << "Not Balanced";
  return 0;
}
```

VIVA QUESTIONS

LAKSHMI NARAIN COLLEGE OF TECHNOLOGY & SCIENCE, BHOPAL

CS-501

Theory Of Computation

1. **Explain Finite automata?**

....
....
....

2. **Design a NFA to accept the string 1001001**

....

....

....

3. **Explain the mechanism of Turing machine?**

....

....

....

4. **What do you mean by Moore Machine?**

....

....

5. **Construct a Push down automata for the given string 0 0 1 1 1 2 2 2 3 3**

....

....
....