Observation notes

Experiment 1:-

write a C program to simulate a Deterministic Finite Automata (DFA) for the given language representing strings that start with a and end with a

Aim:- To write a C program to simulate a Deterministic Finite Automata.

Algorithm:-

1. Draw a DFA for the given language and construct the transition table.
2. store the transition table in a two-dimensional array.
3. Initialize present_state, next_state, final_state
4. Get the input string from the user.
5. Find the length of the input string.
6. Read the input string character by character.
7. Repeat step 8 for every character.
8. Refer the transition table for the entry corresponding to the present state and the current input symbol and update the next state.
9. when we reach the end of the input, if the final state is reached, the input is accepted. Otherwise the input is not accepted.

Program:-

```
#include<stdio.h>
# include<string.h>
# define max 20
int main()
 {
```

```
int trans_table [4] [2] = {{1,3},{1,2},{1,2},{
int final_state=2,i;
int present_state=0;
int next_state=0;
int valid= 0;
char input_string [max];
printf ("Enter a string");
scanf ("%s", input_string);
int l = strlen (input_string);
for(i=0; i<l;i++)
{
   if (input_string[i]=='a')
   next_state=trans_table [present_state] [0];
   else if (input_ string[i]=='b' ]
   next_state= trans_table[Present_state][1];
   else
   invalid =1;
   Present _state = final_State;
}
if (invalid ==1)
{
   Printf ("invalid input");
}
else if (Present_ state== final_state)
Print f ("Accept\n");
else
print f("Don't accept\n");
}.
```

output:-

```
Enter a string: abaab
Accept.
```

Experiment 3:-

Aim:- To write a C program to simulate a Non-Deterministic Finite automata with ε-moves.

Program:-

```
#include <stdio.h>
# include <string.h>
int trans_table [10][5][3];
char symbol [5], a;
int e_closure [10][10], ptr, state;
void find_e_closure(int x);
int main()
{
  int i,j,k,n, num_states, num_symbols;
  for (i=0; i<10; i++)
  {
    for(j=0; j<5; j++)
    {
      for (k=0; k<3; k++)
      {
        trans_table [i][j][k]=-1;
      }
    }
  }
  Printf ("How many states in the NFA with e-moves:");
  scanf ("%d", &num_states);
  Printf ("How many symbols in the input alphabet
          including e:");
  Scanf ("%d", &num_symbols);
```

```
Printf(" ~~How many~~ Enter the symbols without
            Space. Give 'e' first:");
Scanf ("%s", Symbol);
for (i=0; i<num_states; i++)
 {
   for(j=0 ; j<num_symbols; j++)
    {
    Printf ("How many transitions from state %c
            for the input %c:", i, symbol[j]);
    Scanf ("%d", &n);
    for (k=0; k<n ; k++)
   printf("Enter the transitions %d from state %d
            the input %c:", k+1, i, symbol[j]);
   scanf ("%d", &trans_table [i][j][k]);
      }
   }
 }
 for (i=0; i<10; i++)
 {
   for (j=0; j<10; j++)
    {
      e_closure[i][j] = -1;
     }
   }
  for (i=0; i<num_states; i++)
  e_closure [i][0] = i;
  for (i=0; i<num_states; i++)
  {
    if (trans_table [i][0][0] == -1)
    Continue;
    else
     {
      state = i;
       ptr = 1;
      find_e_closure(i);
} }
for (i=0; i<num_states; i++)
{
  printf ("e-closure (%d) = {",i);
  for (j=0; j<num_states; j++)
  {
    if(e_closure [i] [j] != -1)
    {
      printf ("%d," e_closure [i] [j]);
    }
  }
  printf ("}\n");
}}
void find e_closure (intx)
{
  int i, j, y[10], num_trans;
  i=0;
  while (trans_table[x] [0] [i] != -1)
  {
     y [i] = trans_table [x] [0] [i] ;
     i=i+1;
  }
  num_trans=i;
  for (j=0; j<num_trans; j++)
  {
    e_closure [state][ptr]= y[j];
    ptr++;
    find_e_closure(y[j]);
  }}.
```

Output:-

```
How many states in the NFA with e-moves: 3
How many symbols in the input alphabet including e:3
How many transitions from state 1 for the input 1:0
How many transitions from state 2 for the input e:0
How many transitions from state 2 for the input 0:0
How many transitions from state 2 for the input
How many transitions from state 2 for the input
                                            1 : 0
```

ε-closure(0)= {0,1,2,}

ε-closure(1)= {1,2,}

ε-closure(2)= {2,}

Experiment 3:-

Aim:- To write a C program to check whether a string belongs to the grammar

S → 0A1

A → 0A|1A|ε

Program:-

```
#include <stdio.h>
#include <string.h>
int main(){
 char s[100];
 int i, flag;
 int l;
 printf("Enter a string to check:");
 scanf("%s", s);
 l = strlen(s);
 flag=1;
 for(i=0; i<l; i++)
 {
     if(s[i]!='0' && s[i]!='1')
     {
      flag=0;
     }}
if(flag!=1)
      printf("string is not valid\n");
if(flag==1){
      if(s[0]=='0' && s[l-1]=='1')
          printf("string is accepted\n");
      else
          printf("string is not accepted\n");
      }}
```

output:-

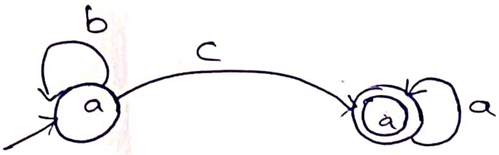
Enter a string to check: 0101011101

string is accepted.

Experiment 4:-

Design DFA to accept bcaaaaaaaaa, bc and c

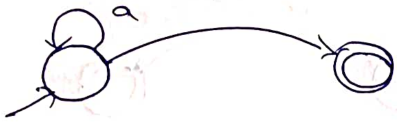
DFA:-



b	c	a	a	a	a	a	a	a	a	▽	

Experiment 5:-

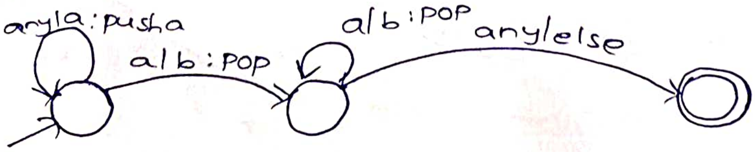
Design NFA to accept aaaaaa



a	a	a	a	a	a	a	a	a	a	a	a	▽

Experiment 6:-

Design PDA for the input a^n b^n

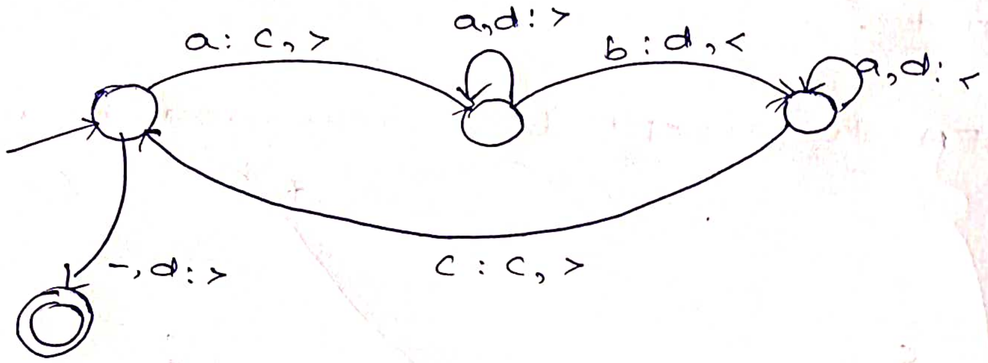


	a	a	b	b					

Experiment 7:-

Design TM for input a^n b^n

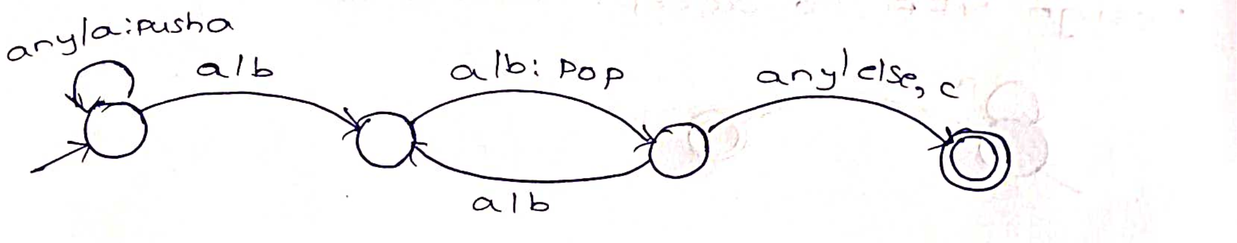
TM:-



		c	c	d	d			

Experiment 8:-

Design PDA for input aabbbbc (L = a^n b^2n)

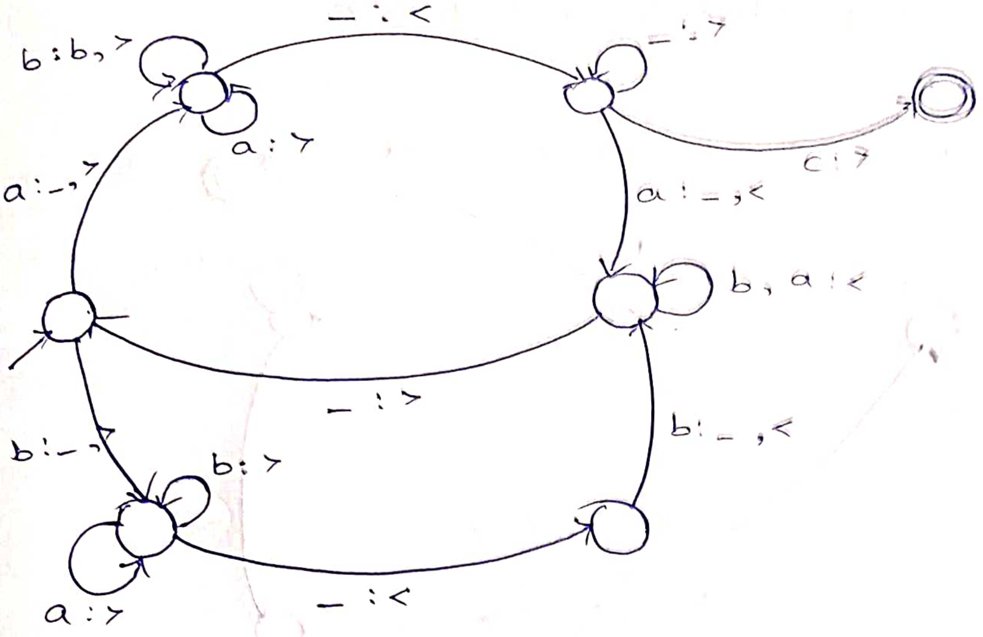


	a	a	b	b	b	b		

Experiment 9:-

TM simulation for palindrome w = abab c

TM:-



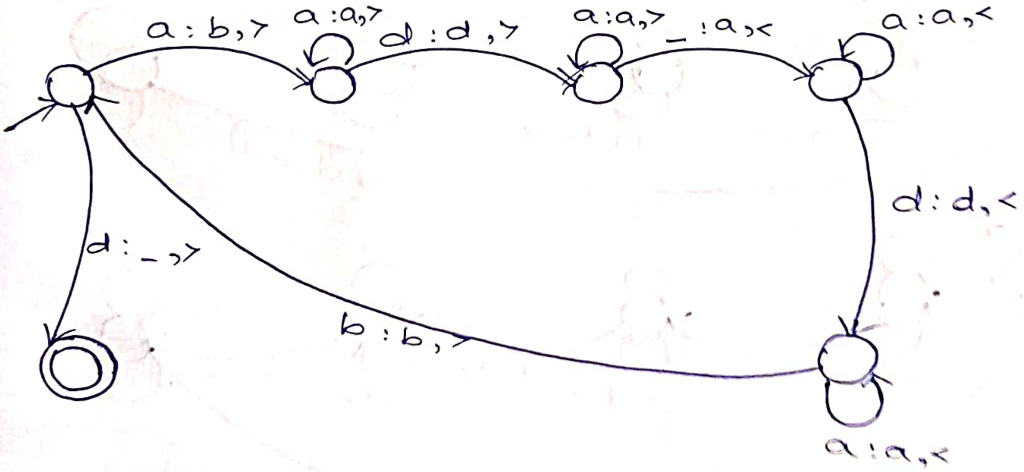
	▽ a	b	a	b	a		c			

Experiment 10:-

Design TM to perform addition of following

w= aa + aaaa

After Addition of a's = aaaaaa



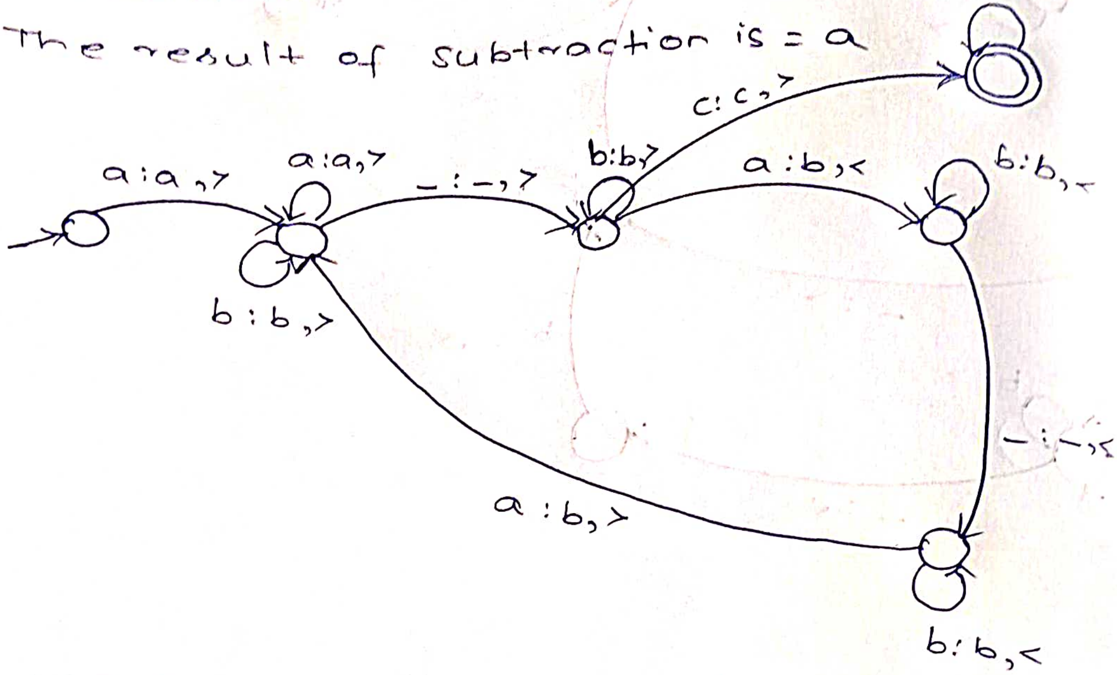
	b	b		▽ a	a	a	a	a	a	

Experiment 11:-

Design Tm to perform subtraction

w = aaa-aa

The result of subtraction is = a

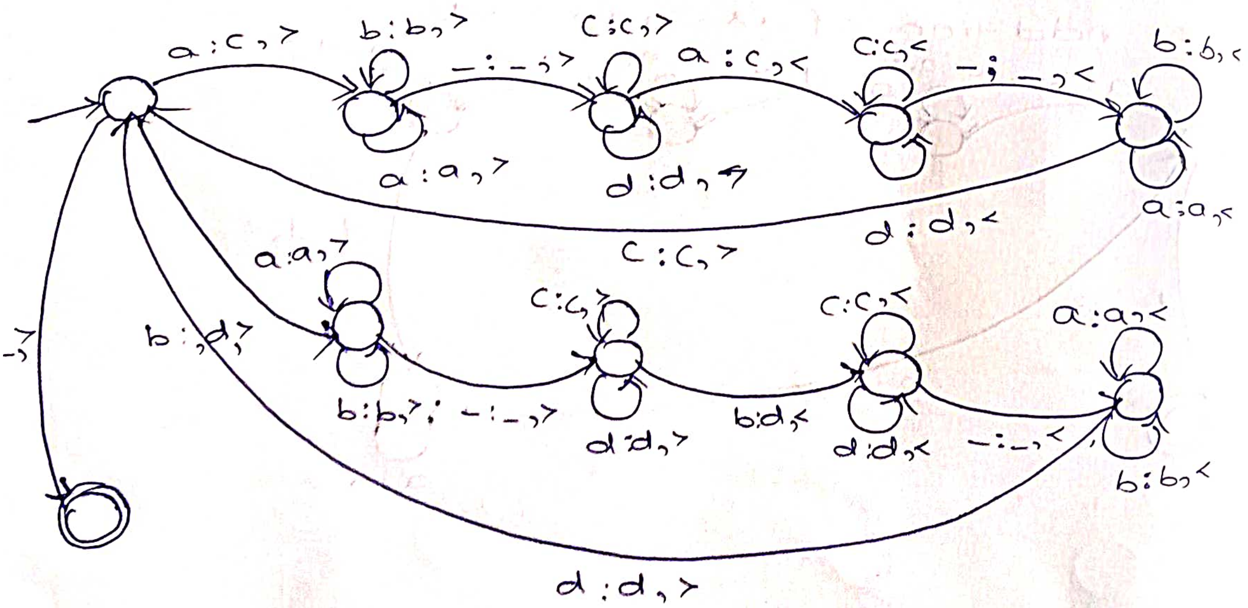


a	b	b		b	b	C	▽			

Experiment 12:-

Design Tm to perform string comparison

w = aba aba



c	d	c		▽ c	d	c			

Experiment 13:-

Aim:- To write a C program to simulate a Non-Deterministic Finite Automata.

Program:-

```
# include <stdio.h>
# include <string:h>
int main()
{
 int i,j,k,l,m, next_state[20],n,mat[10][10][10],flag,p;
 int num_states,final_state[5],num_symbols,num_final
 int present_state[20], prev_trans, new_trans;
 char ch, input[20];
 int symbol[5], inp,inp1;
 printf("How many states in the NFA:");
 Scanf("%d",&num_states);
 printf(" How many symbols in the input alphabet:",
 Scanf("%d",& num_symbols);
 for(i=0; i<num_symbols; i++)
 {
    printf("Enter the input symbol%d:",i+1);
    Scanf("%d",&symbol[i]);
 }
 printf("How many final states:");
 Scanf("%d",&num_final);
 for(i=0;i<num_final;i++)
 {
    printf("Enter the final state %d:",i+1);
    Scanf("%d",& final_state[i]);
 }
 for(i=0; i<10; i++)
 {
    for (j=0; j<10; j++)
    {
       for(k=0;k<10;k++)
```

```
        {
            mat[i][j][k] = -1;
          }
      }
  }
  for (i=0; i<num_states ; i++)
   {
       for (j=0; j<num_symbols; j++)
       {
         printf ("How many transitions from state
          %d for the input %d: ", i, symbol[j]);
          Scanf("%d",&n);
          for (k=0; k<n; k++)
          {
        printf("Enter the transition %d from state
          %d for the input %d:", k+1, i, symbol[j]);
         Scanf ("%d", & mat [i][j][k]);
       }}}
    printf ("The transitions are stored as shown
                  below\n");
   for (i=0; i<10; i++)
     {
       for (j=0; j<10; j++)
         {
           for (k=0; k<10; k++)
             {
               if (mat[i][j][k]! =-1)
    printf( "mat [%d] [%d][%d]=%d\n", i, j ,k mat[i][j][
         }}}
   while(1)
    {
     printf ("Enter the input string: ");
```

```
scanf ("%s", input);
 present_state [0] = 0;
  prev_trans = 1;
   l = strlen (input);
   for (i=0; i<l; i++)
   {
     if (input[i]=='0')
 inp1=0;
 else if (input[i]=='1')
 inp1 = 1;
 else
  {
  printf ("invalid input\n");
  exit(0);
  }
 for (m=0; m<num_symbols; m++)
    {
        if (inp1 == symbol [m])
         {
            inp= m;
            break;
          }}
        new_trans=0;
        for (j=0; j<prev_trans, j++)
        {
            k= 0;
             p= present_state [j];
        while (mat [p] [inp][k]! =-1)
        {
         next_state [new_trans++] =mat[p][inp][k];
          k++; } }
   for (j=0; j<new_trans; j++)
   {
     for (j=0 ; j< num_final; j++)
       {
          if (present_state[i]== final_state[j])
{
    flag=1;
    break;
    }}}
  if (flag==1)
     printf ("Accepted\n");
  else
      printf ("Not accepted \n");
   printf ("Try with another input\n");
    }}
```

Out put:-

```
Enter the input string: 0111010
 Accepted
Try with another input.
Enter the input string: 10010101
Accepted
Try with another input
 Enter the input string: 100100
 Not accepted.
 Try with another input
  Enter the input string: 011011
  Not accepted.
```