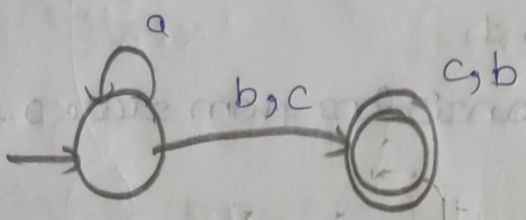
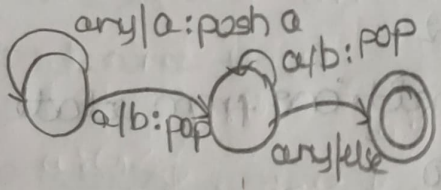
SIMULATORS

12) Design DFA using simulator to accept the input string "a", "ac", and "bac"

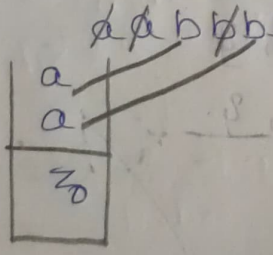


13) Design PDA using simulator to accept the input string aabb.



Input : aabb

14) Design PDA using simulator to accept the input string $a^n b^n$



$\delta(q_0, q, z_0) = (q_1, az_0)$

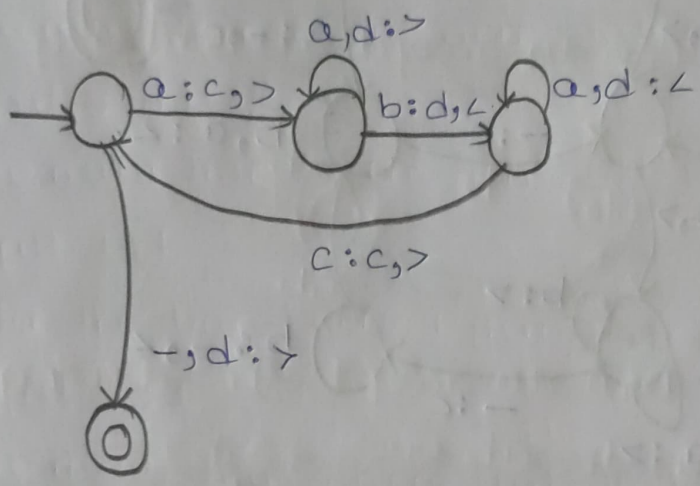
$\delta(q_1, a, a) = (q_1, aa)$

14) $\delta(a,b,a) = (q_2, aa)$

$\delta(q_2,b,a) = (q_3, \lambda)$

$\delta(q_3, \lambda, z_0) = (q_4, z_0)$

15) Design TM to accept the input string $A^n b^n$



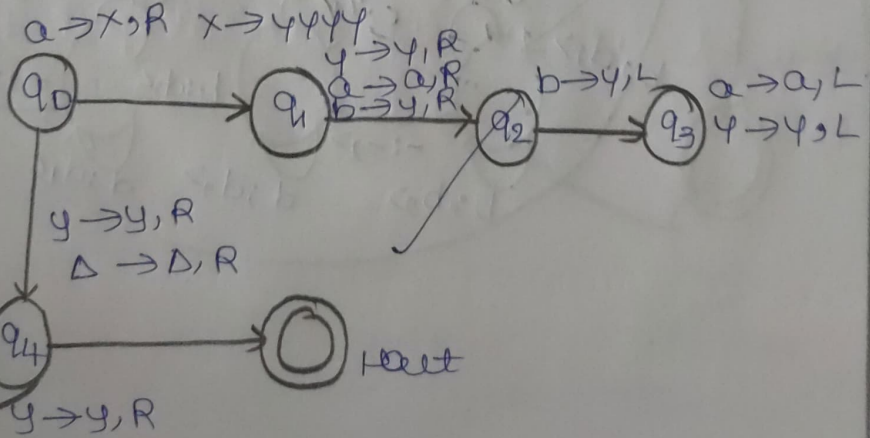
16) Design TM to accept input string $A^n B^{2n}$

aa bb bb

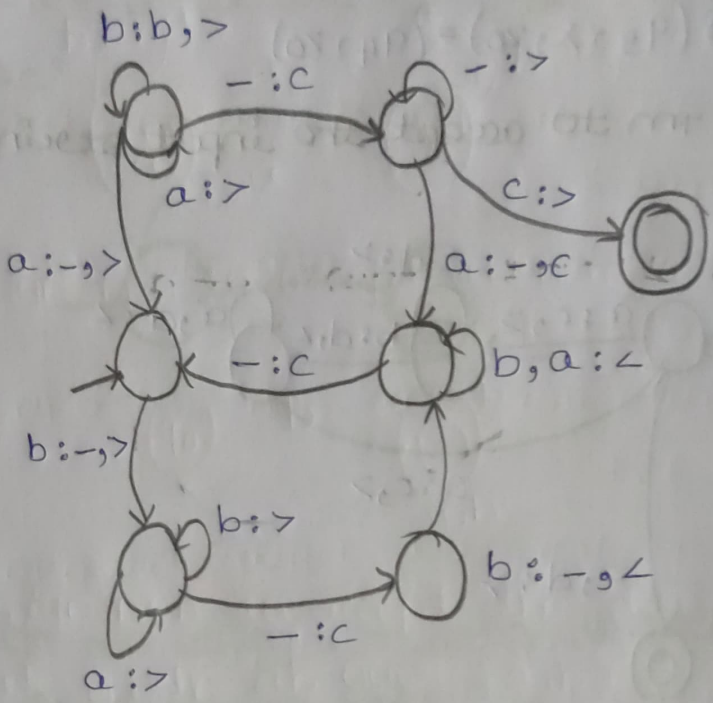
xa yy ←

x yy yy ←

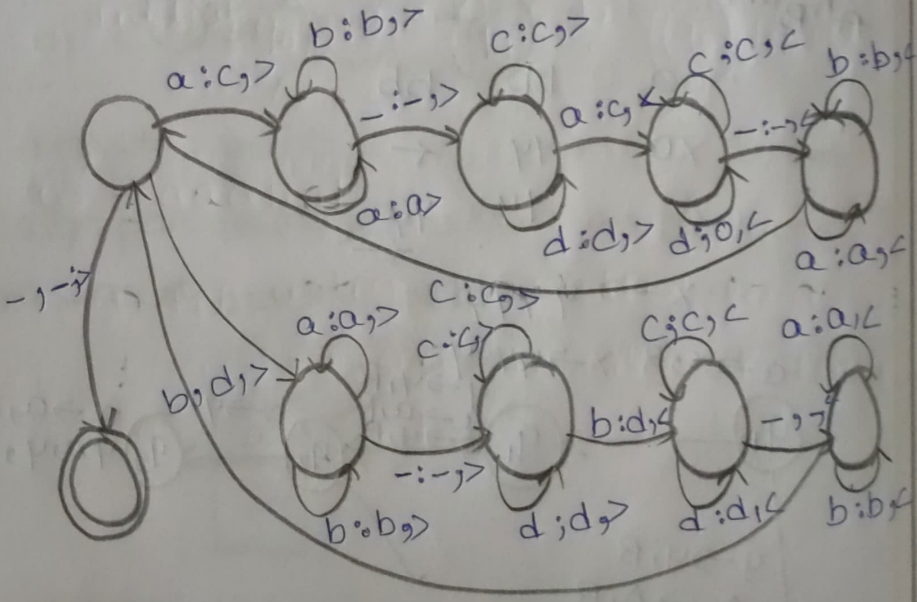
x → yyyy



17) Design TM using simulator to accept the input string palindrome ababa



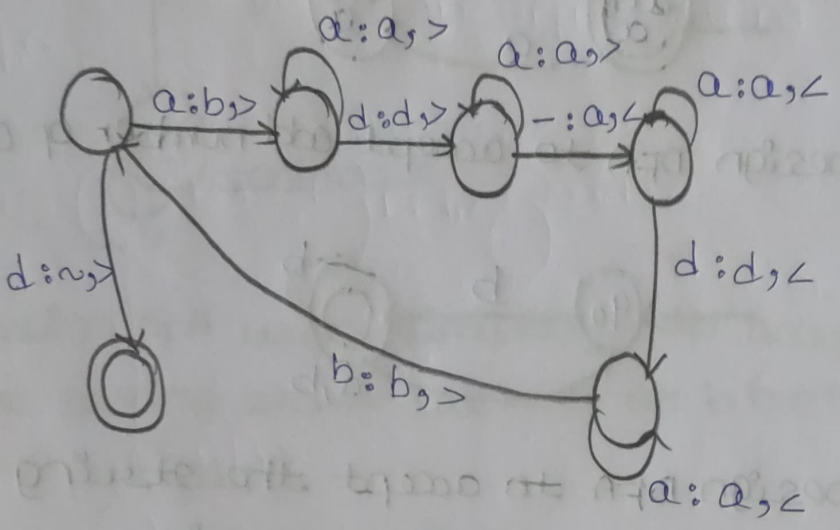
18) Design TM using simulator to accept input string ww



19) Design TM using simulator to accept perform addition of "a' of 'aaa'

$w = aa + aaaa$

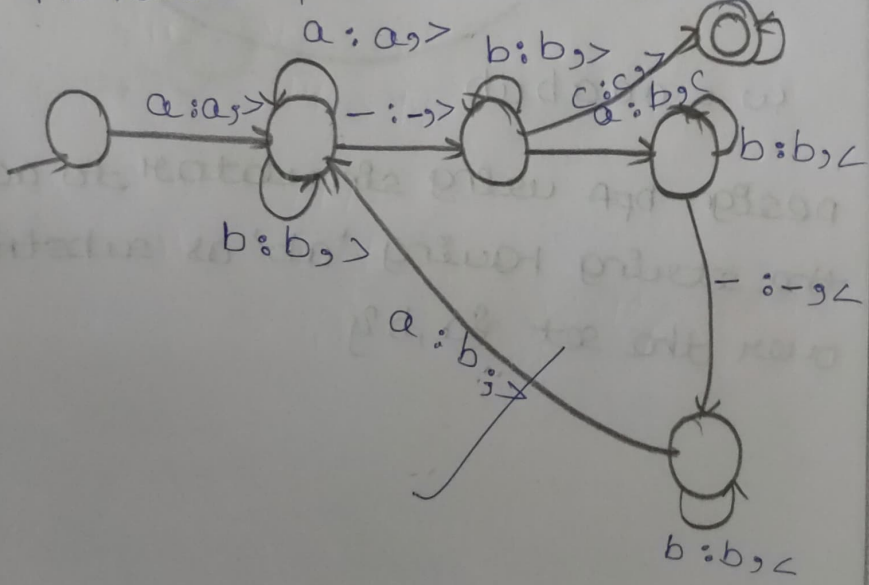
After addition of a's = aaaaaaa



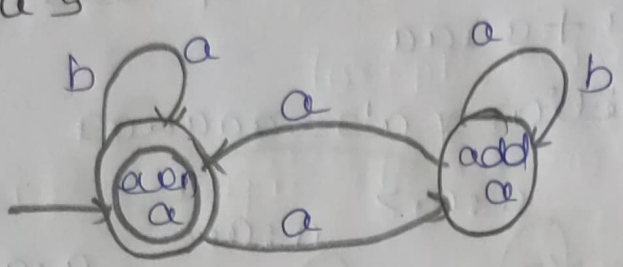
20) Design TM using simulator to perform subtraction of 'aaa' and 'aa'

$w = aaa - aa$

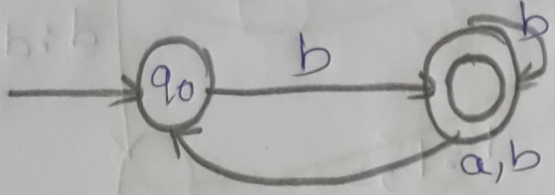
the result of subtraction is = a



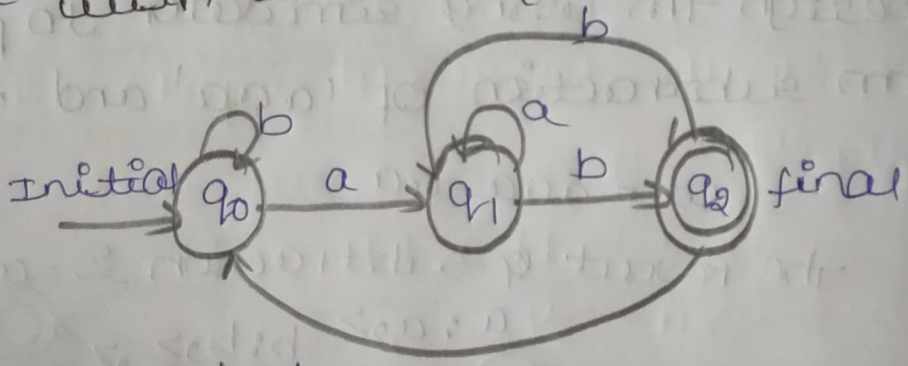
21) Design DFA to accept even number of a's



22) Design DFA to accept odd number of a's

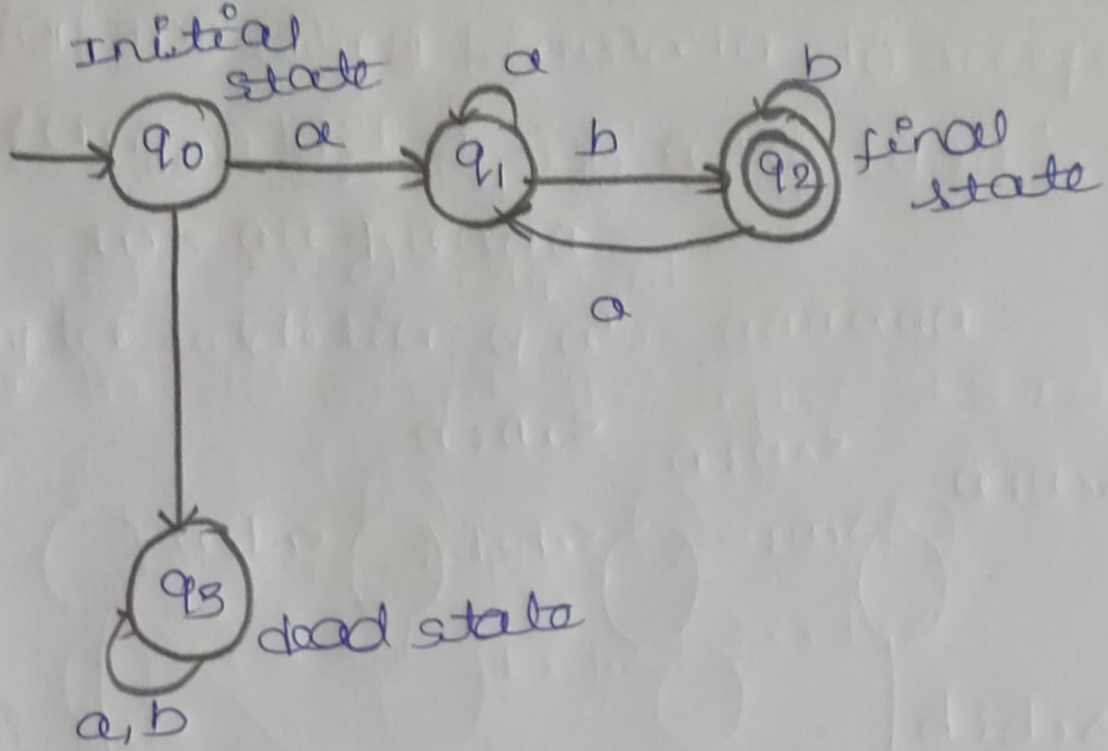


23) Design DFA to accept the string that with ab over {a,b}

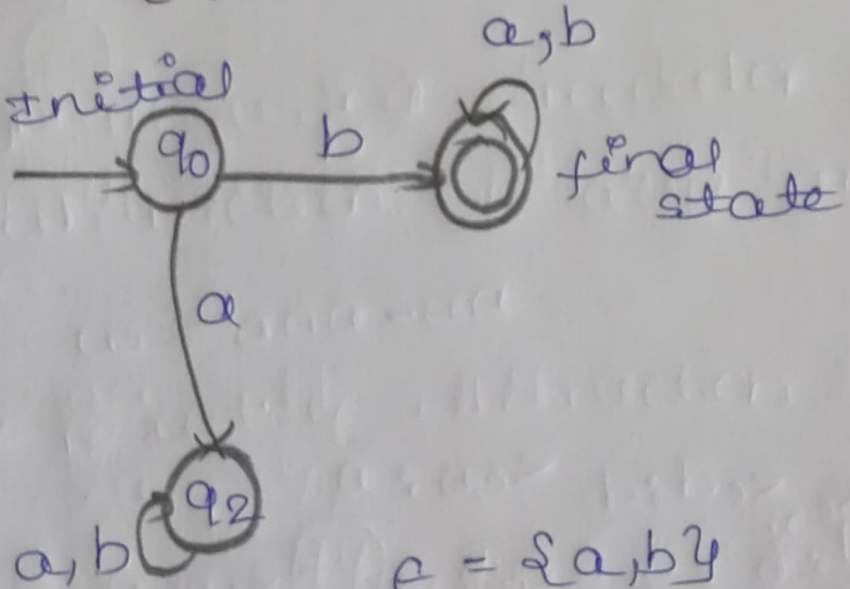


$w = aaabab$

24) Design DFA using simulator to accept the string having 'ab' as substring over the set {a,b}



25) Design DFA using simulator to accept the string start with a or b over the set {a,b}



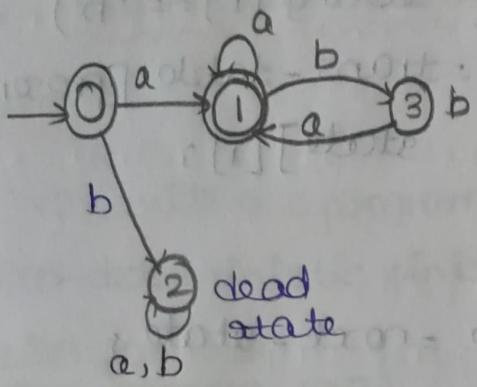
$\epsilon = \{a, b\}$

1) write a c program to stimulate a deterministic finite Automata (DFA)

AIM

TO write a c program to stimulate a Deterministic finite Automata.

Design for DFA



program

```
#include <stdio.h>
#include<string.h>
#define max 20
int main()
{
int trans_table [4][2]={{1,3},{1,2},
                        {1,2},{3,3}};
Int final_state =2,i;
int present_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_sta
                          te][0];
 else if(input_string[i]=='b')
 next_state = trans_table[present_
                          state][1];
 else
 invalid = 1;
 present_state = next_state;
}
if(invalid==1)
{
 printf("Invalid input");
}
else if(present_state == final state)
printf("Accept\n");
else
printf("don't accept\n");
}
```

output

Enter a string : abaab

Accept

Result

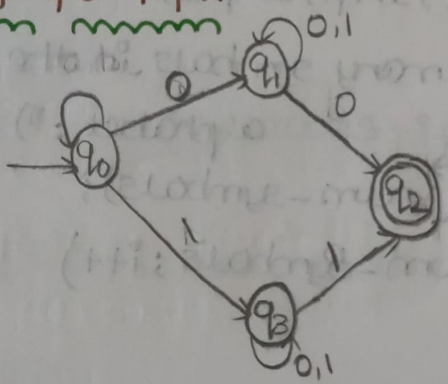
thus the given c program for DFA is executed successfully

Non-Deterministic finite Automata

AIM

TO write a c program to stimulate a Non-deterministic finite automata.

Design for NFA



transition table

state/input	0	1
→0	1	3
1	{1,2}	1
2	-	-
3	3	{2,3}

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 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] [k]);
    }
  }
}
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++)
```

```
{
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++)
{
present_state[j]=next_state[j];
}
prev_trans=new_trans;
}
```

```
flag=0;
for(i=0;i<prev-trans;i++)
{
for(j=0; j<num_final;j++)
{
if(present_state[i]==final_state[j])
{
flag=1;
break;
}
}
}
If(flag==1)
print("Not accepted \n");
printf("try with another input \n");
}
}
```

output

How many states in the NFA : 4
How many symbols in the input alphabet: 2
Enter the input symbol 1 :0
Enter the input symbol 2 : 1
How many final states : 1
Enter the final state 1 : 2
How many transitions from state 0 for the input 0: 1
Enter the transition 1 from state 0

for the input 0 : 1
How many transitions from state 0 for the input 1 : 1
Enter the transitions 1 from state 0 for the input 1 : 3
How many transitions from state 1 for the input 0 : 2
Enter the transition 1 from state 1 for the input 0 : 1
Enter the transitions 2 from state 1 for the input 0 : 2
How many transitions from state 1 for the input 1 : 1
Enter the transition 1 from state 1 for the input 1 : 1
How many transitions from state 2 for the input 0 : 0
How many transitions from state 3 for the input 0 : 1
Enter the transition 1 from state 3 for the input 1 : 2
Enter the transition 2 from state 3 for the input 1 : 3
the transitions are stored as shown below
mat [0][0][0] = 1
mat [0][1][0] = 3

```
mat[1][0][0] = 1
mat[1][0][1] = 2
mat[1][1][0] = 1
mat[3][0][0] = 3
mat[3][1][0] = 2
mat[3][1][1] = 3
Enter the input string : 0111010
Accepted
Try with another input
Enter the input string : 100100
Not accepted
Try with another input
```

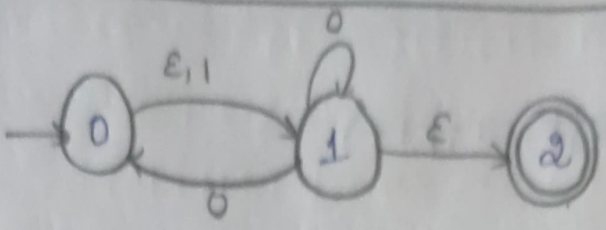
Result

Thus the given c program for NFA is executed successfully.

3. Finding ε-closure for NFA with ε-moves

AIM

To write a c program to find ε-closure of a non-deterministic finite Automata with ε-moves.



Diagram

Transition table

state / input	ε	0	1
→0	1	-	1
1	2	{0,1}	-
②	-	-	-

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_states);
printf("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
      %d for the
scanf("%d",&n);
for(k=0;k<n;k++)
{
printf("Enter the transitions %d from
            %d for the input %c:",k+1)
```

```
scanf("%d", &trans_tabb[i][j][r]);
}
}
}
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_tabb[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]);
}
}
print("\n");
}
}
void find_e_closure(int x)
{
int i, j, y[10], num_trans;
i=0;
while (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

Enter the symbols without space. Give 'e'.
first : e01
How many transitions from state 0 for the input e : 1
Enter the transitions 1 from state 0 for the input e: 1
How many transitions from state 0 for the input 0 :0
How many transitions from state 0 for the input 1 : 1
Enter the transitions 1 from state 0 for the input 1 : 1
How many transitions from state 1 for the input 0 : 2
How many transitions from state 2 for the input 0 :0
How many transitions from state 2 for the input 1: 0
e-closure (0) = {0, 1, 2}
e-closure (1) = {1, 2, }
e-closure (2) = {2, }

Result

Thus, the given c program e-closure for NFA is executed successfully