Deterministic finite Automata (DFA)

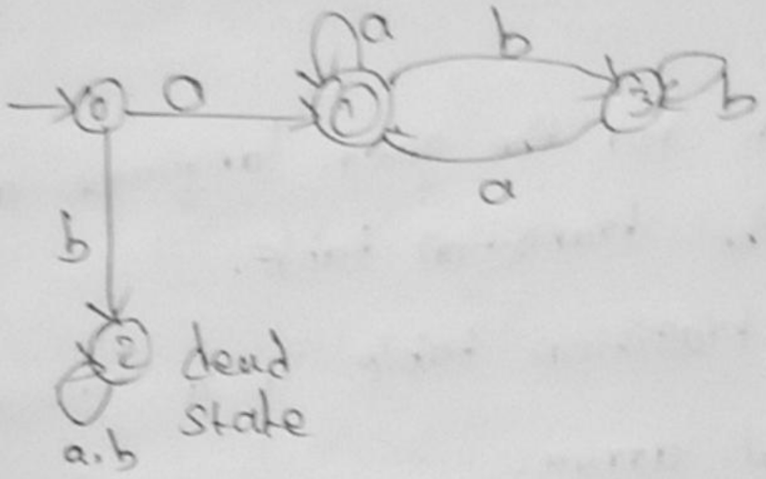
Ex:-1

Aim: To write a C-program to simulate a deterministic finite Automata.

Algorithm:

- * Draw a DFA for the given language and construct the transition table.
- * Store the trasition table in a two-dimensional array.
- * . Initialize present-state; next-state and final state.
- * Get the input from the user.
- * Find the length of input string.
- * Read the input string character,
- * Repeat step 8 for every character.
- * Refer the transition table for the entry corresponding to the present state and current input symbol. and update the next state
- * when we reach the end of input, if the final state is reached, the input is accepted.

Ex Simulate a DFA for language representing strings over Σ = {a,b} that start with a and end with b



Transition table:

Present State	Next State a	b
0	1	2
1	1	3
2	2	2
3	1	3

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 next - state =0;
int invalid =0;
char input - String {max};
Print {"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;
 }
 if (invalid ==1)
 {
```

```
{
Printf ("input invalid");

}
else if (present-state::final state)
Printf ("Accept\n");
else
printf ("Don't Accept\n");
```

output :-

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

Result: Thus the C-Program was done and implemented successfully.

(2) checking whether a String belongs to a grammar

Aim: To write a C-Program to check whether a String belongs to grammar

S->0A1

A->0A|1A|ε

language defined by the Grammar:

Set of all String over Σ=(0,1) that Start with 0 and end with 1

Algorithm:

1. Get the input String from the user.
2. find the length of the String.
3. check whether all the Symbol in the put are either 0 or 1. If So print "String is valid" and go to Step 4.
4. If the first Symbol is 0 and last Symbol is 1 Print "String accepted", otherwise printf "String not accepted"

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: 1

```
enter a String to check = 01010111101
 String is accepted
```

output : 2

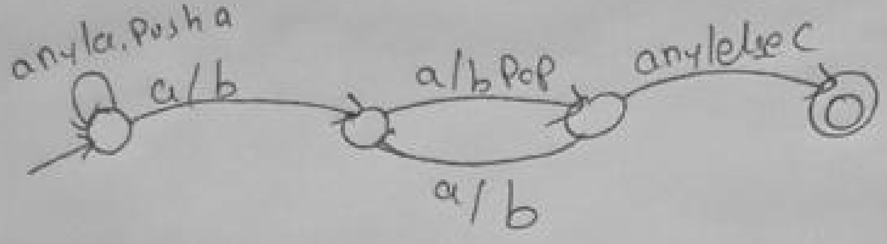
```
enter a String to check =011101010110
String is not,accepted
```

output : 3

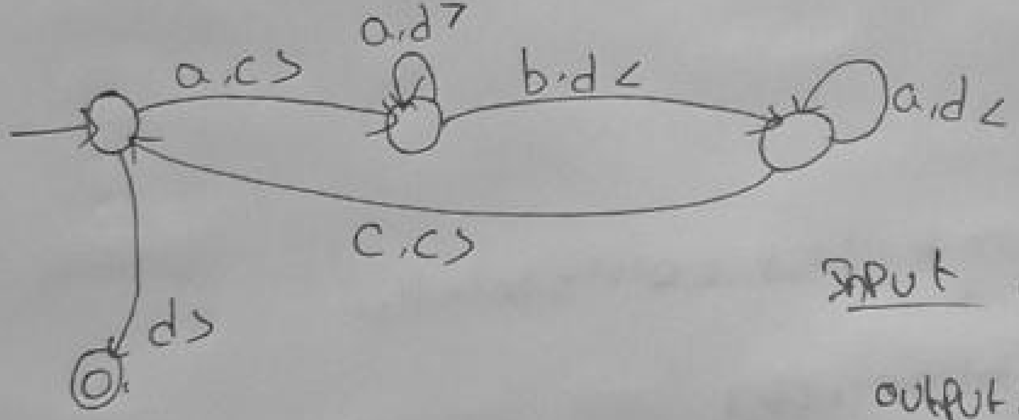
```
enter a String to check : abbbaaba
String is not valid.
```

Result: Thus the program was implemented and done Sucessfully.

① Design PDA using simulator to accept the input String a^nb^2n



② Desing TM using simulator to accept the input string A^nB^2n



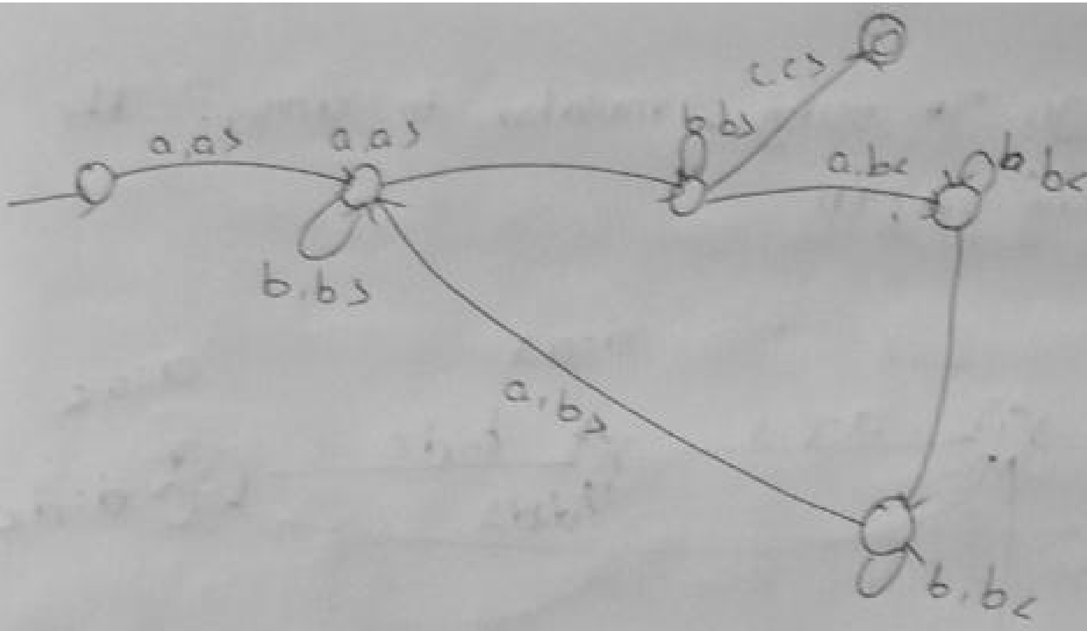
Input aabb

output: ccdd.

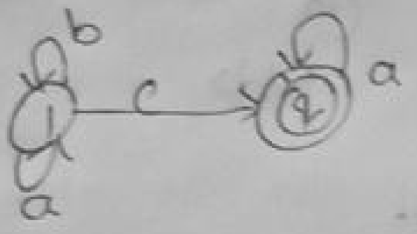
③ Design TM using Simulator to Perform substraction of aaa-aa

W=aaa-aa

The Result of substraction is =a



4 Design DFA using simulators to accept the input string 'a' 'ac' and bac .



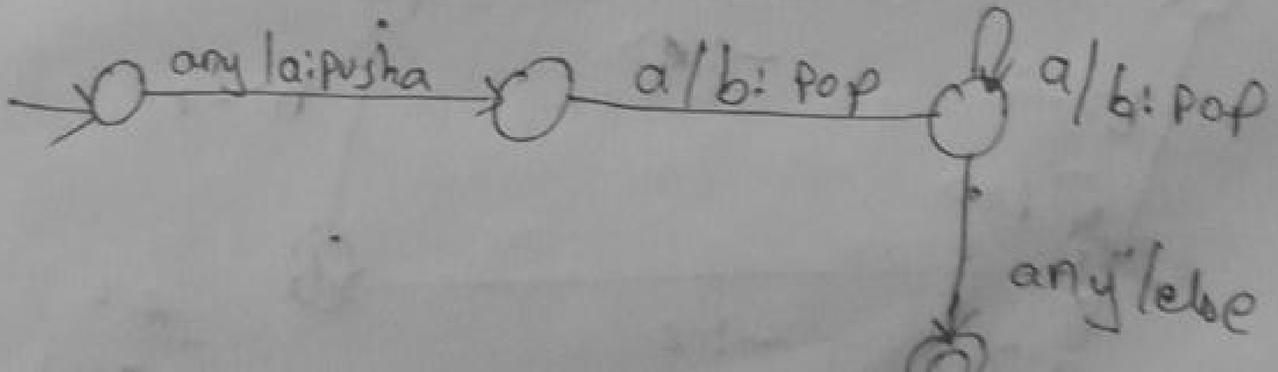
output

→ bcaaaa

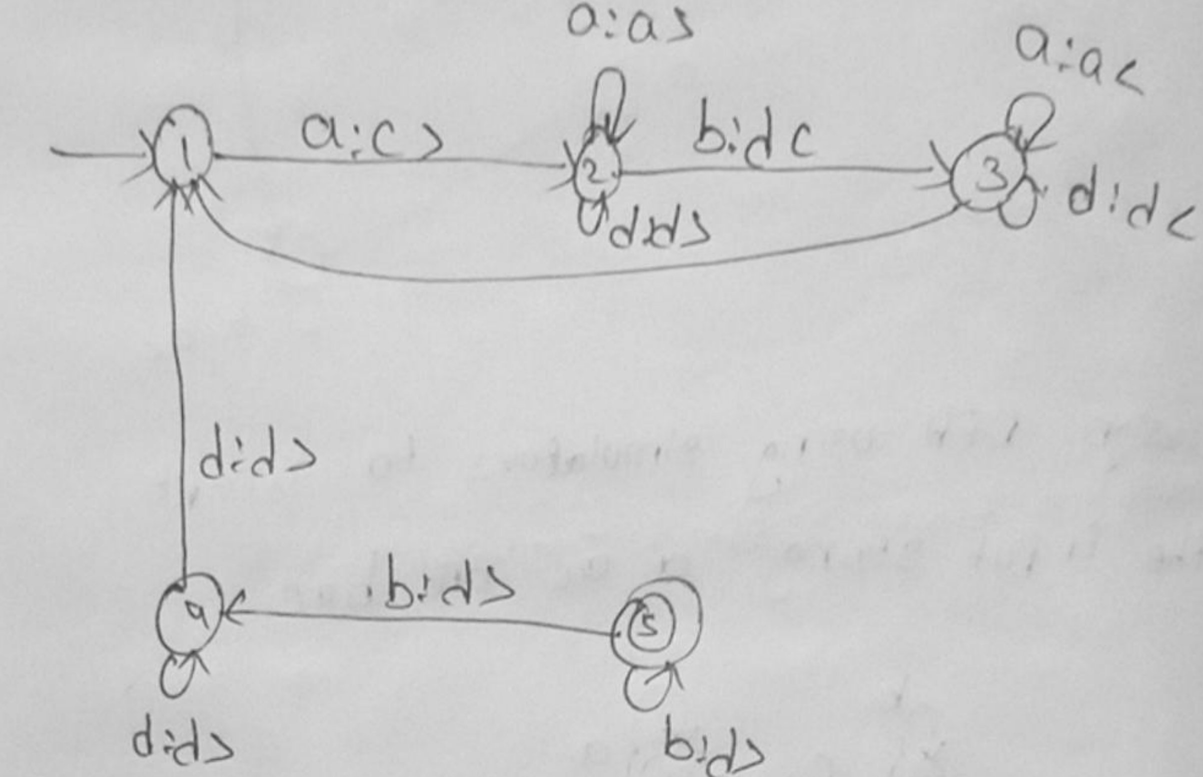
→ bc

→ a

5 Design PDA using simulator to accept the input string aabb



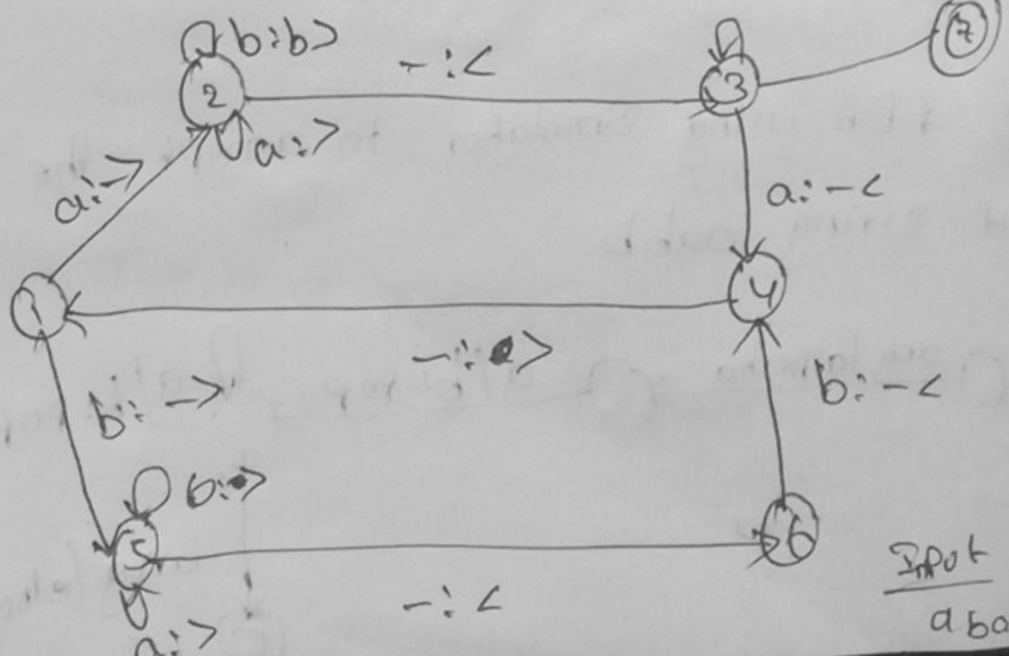
Design Tm using simulator to accept the String $a^n b^{2n}$



Input
aaabbbbbb

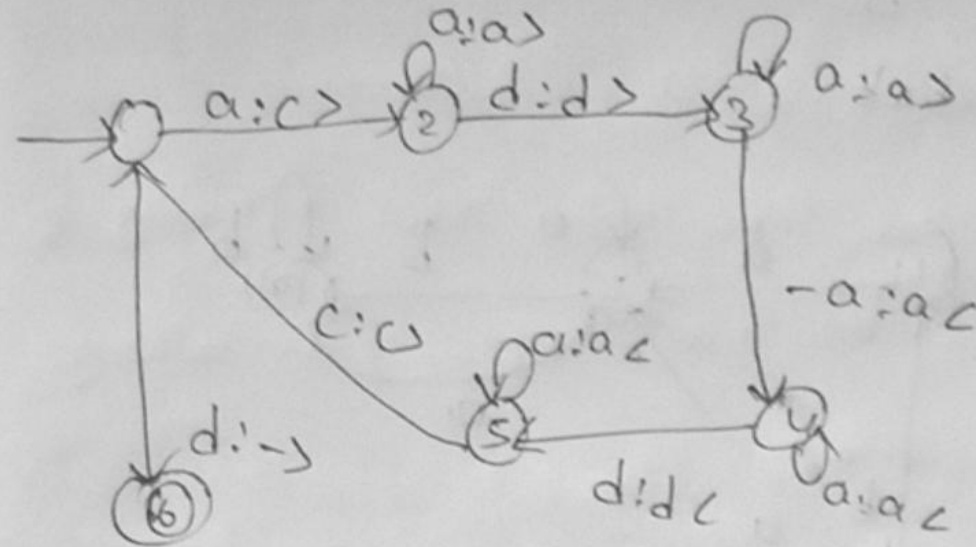
output
cccdddddd

Design Tm using simulator to accept input String palindrome ababa

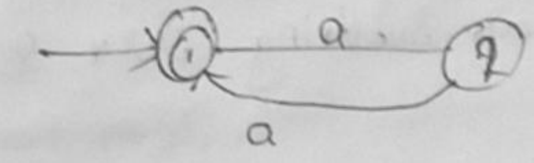


Input
ababa

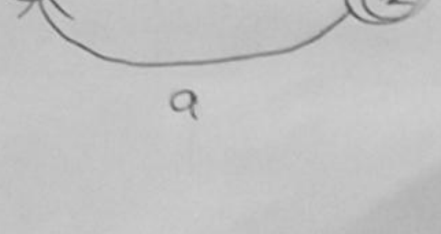
Design Tm using Simulator to perform addition of ba and aaa



Design DFA using simulator to accept even number a's

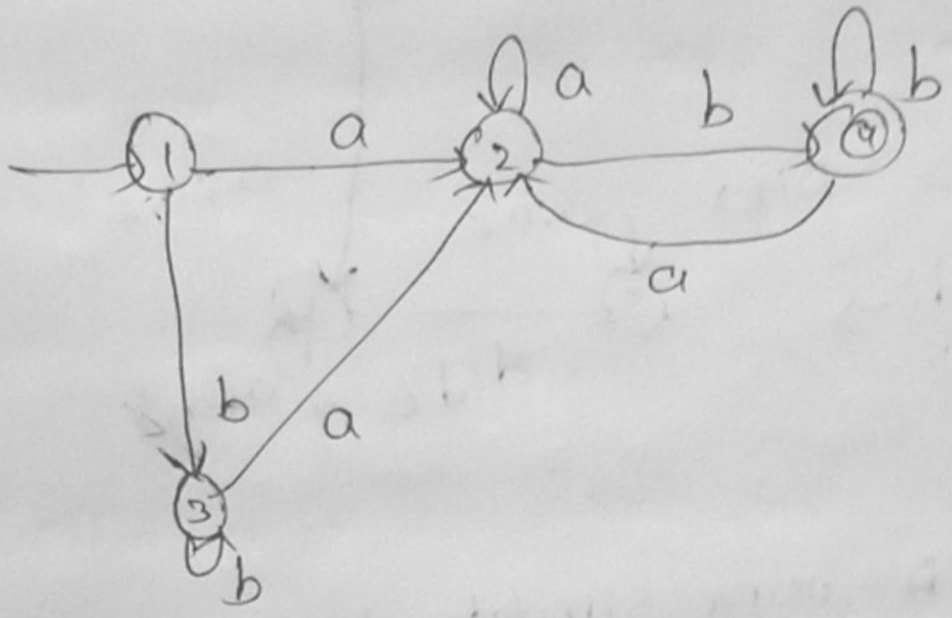


Design DFA using Simulator to accept odd number a

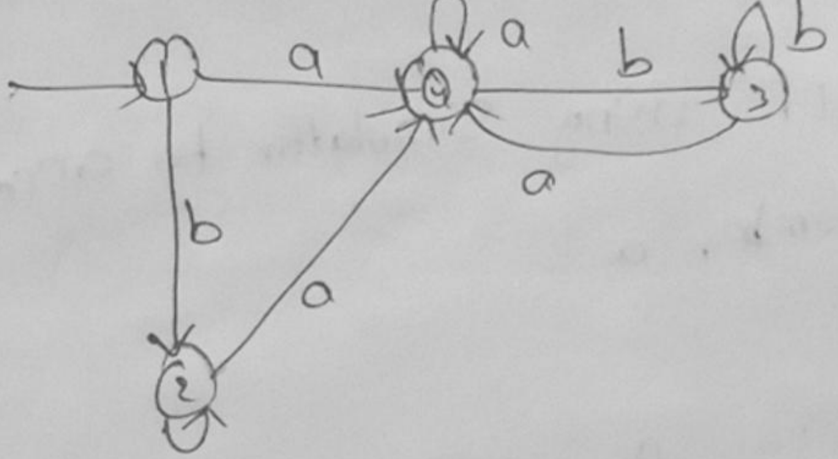


Design DFA using simulator to accept the String the end with ab over set {a,b}

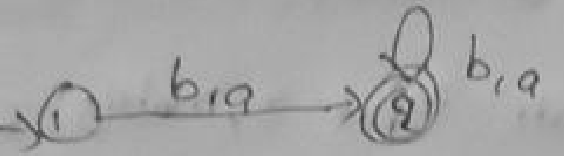
w = aabab



Design DFA using Simulator to accept the string having ab as Substring over set {ab}



Design DFA using simulator to accept the String Start with a (or) b over the set (a,b)



Finding ε-closure for NFA with ε-moves

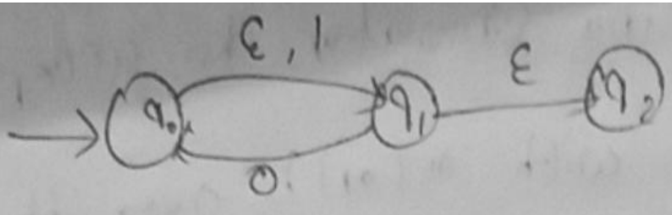
Aim: ~~To~~ To write a C-program to find ε-closure of a Non-Deterministic Finite Automata with ε-moves.

Algorithm:

1. get the following as input from the user.

2. Declare a 3-dimensional matrix to store the transitions and initialize.

3. Get the transitions from every state for every input symbol from user.

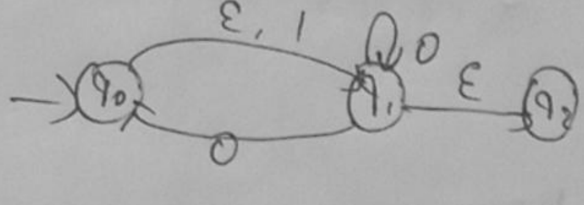


There are 3 states 0, 1, and 2

There are three state input symbols ε, 0 and 1. As the array index always starts with 0.

4. Initialize a two dimensional matrix e-closure with -1 in all the entries

5. ε-closure of a state q is defined as set of all states that can reached from state q using only ε-transitions.



ε-closure (0) = (0, 1, 2)
ε-closure (1) = {1, 2}
ε-closure (2) = {2}

6. for every State p, find ε-closure as follows.

7. for every State Print ε-closure values!

Program:

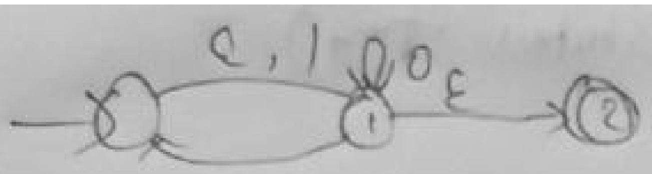
```
#inclde <stdio.h>
# include < string.h >
  char symbol{5],a:
int e-closure [10][10], ptr, state;
void find e-closure [int x)
 int main ( )
 {
 int i,j k  num-state, num-symbols.
   for ( i=0: i<10: i++)
  {
    for (j=0 ;j< 5;j++)
    {
      for (k=0; k<3;k++)
     { trans-table [i][j][k]=-1
```

```
      }
    }
  }
Printf ("How many States in NFD with e
 Scant ("%d", &num-States);
Printf ("How many Symboles e.: ");
  Scanf ("%S", Symbols);
for(i=0:i<num-states;i++).
 {
  for (j=0; j<num-symbols: j++)

 {
   Scanf ("%d", &n)
   for (k=0: k<n;k+)
     }
   }
 }
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);
    }

 }
e-closure [state [ptr]: y[j];
    ptr++;
  } find e-closure (y[j])
 }
```

example

Find ε-closure for all states for NFA with ε-moves given below.



Transition table

Input	ε	0	1
0	1	–	1
1	2	{0,1}	–
2	–	–	–

output

How many states with ε moves: 3

How many symbols in input = 3

e-closure (0) : {0,1,2}

e-closure (1) : {1,2}

e-closure (2) : {2,}