10.Write a C program to find ε -closure for all the states in a Non-Deterministic Finite Automata (NFA) with ε -moves.

Aim: To write a C program to find ε -closure for all the states in a Non-Deterministic Finite Automata (NFA) with ε -moves.

Algorithm:

1. Get the following as input from the user.

i. Number of states in the NFA

ii. Number of symbols in the input alphabet including ε

iii. Input symbols

iv. Number of final states and their names

2. Declare a 3-dimensional matrix to store the transitions and initialize all the entries with -1

3. Get the transitions from every state for every input symbol from the user and store it in the matrix.

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

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

6. For every state i, find ε-closure as follows: If there is an ε-transition from state i to state j, add j to the matrix e\_closure[i]. Call the recursive function find\_e\_closure(j) and add the other states that are reachable from i using ε

7. For every state, print the ε-closure values

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 may 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("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 input %c:",i,symbol[j]);

scanf("%d",&n);

for(k=0;k<n;k++)

{

printf("Enter the transitions %d from state %d for 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(int x)

{

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]);

}

} #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 may 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("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 input %c:",i,symbol[j]);

scanf("%d",&n);

for(k=0;k<n;k++)

{

printf("Enter the transitions %d from state %d for 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(int x)

{

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];

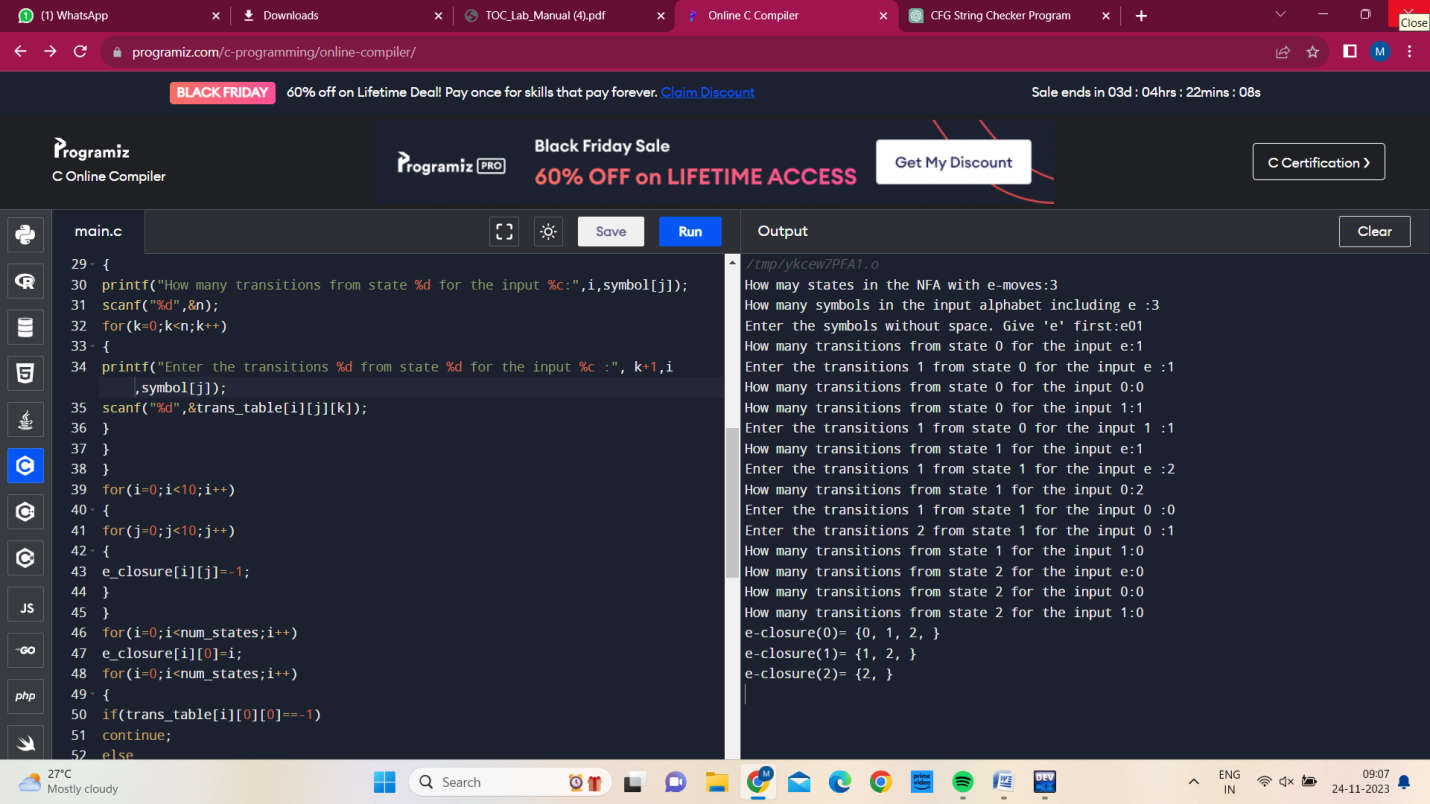
ptr++;

find\_e\_closure(y[j]);

}

}

Output:



Result:

C program is successfully executed.