**Aim: Find Epsilon – Closure For All States In NFA**

**Algorithm:**

**Algorithm**

Initialize a Stack with the Starting State:

* Start with a stack containing only the starting state.
* Build the Epsilon Closure Set:
* Initialize an empty epsilon closure set.
* While the stack is not empty:
* Pop a state from the stack, and call it current\_state.
* If current\_state is not in the epsilon closure set:
  + Add current\_state to the epsilon closure set.
  + If current\_state has epsilon transitions:
    - Add all states reachable via epsilon transitions to the stack.
* Return the Epsilon Closure Set
* Result.

After the stack is empty, the epsilon closure set contains the desired

**Program:**

#include <stdio.h>

#include <stdbool.h>

#define MAX\_STATES 50

#define MAX\_TRANSITIONS 50

typedef struct

{

int state;

int transition[MAX\_TRANSITIONS];

} NFAState;

NFAState nfa[MAX\_STATES];

int numStates;

void epsilonClosure(int state, bool visited[])

{

visited[state] = true;

for (int i = 0; nfa[state].transition[i] != -1; i++)

{

int nextState = nfa[state].transition[i];

if (!visited[nextState])

epsilonClosure(nextState, visited);

}

}

int main()

{

printf("Enter Number of States in NFA: ");

scanf("%d", &numStates);

for (int i = 0; i < numStates; i++)

{

nfa[i].state = i;

printf("Enter Number of Epsilon Transitions For State %d (Enter -1 to End) : ", i);

int numTransitions = 0;

do

{

scanf("%d", &nfa[i].transition[numTransitions]);

numTransitions++;

}

while (nfa[i].transition[numTransitions - 1] != -1);

}

printf("\nEpsilon-Closure For Each State :\n");

for (int i = 0; i < numStates; i++)

{

bool visited[MAX\_STATES] = {false};

epsilonClosure(i, visited);

printf("Epsilon-Closure of State %d : ", i);

for (int j = 0; j < numStates; j++)

if (visited[j])

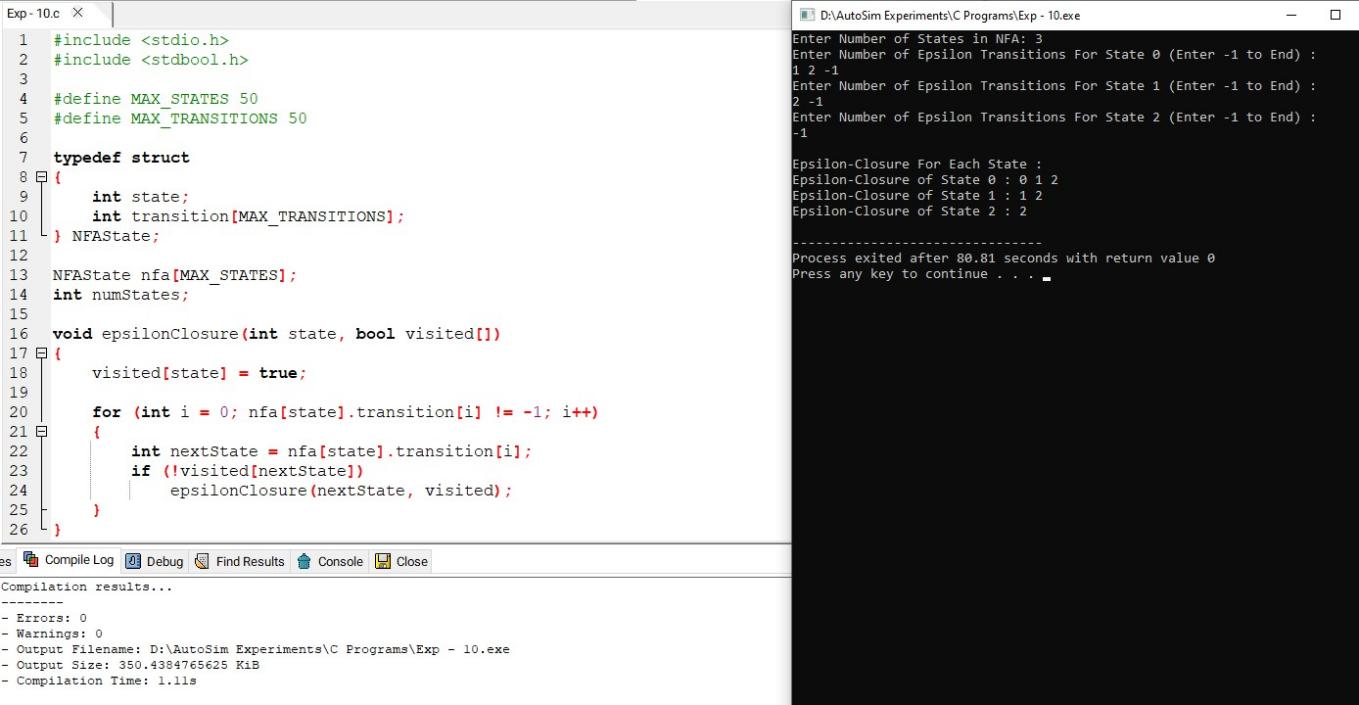
printf("%d ", j);

printf("\n");

}

return 0;

}



**Result**

Thus we have successfully implemented and executed the program and the strings given as inputs are verified