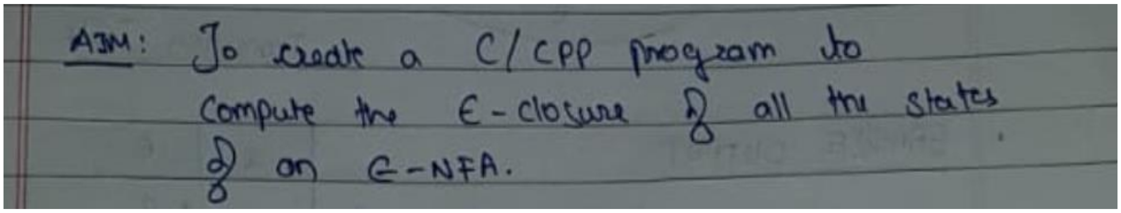
Epsilon Closure Computation using CPP
Compiler Design – Lab 2

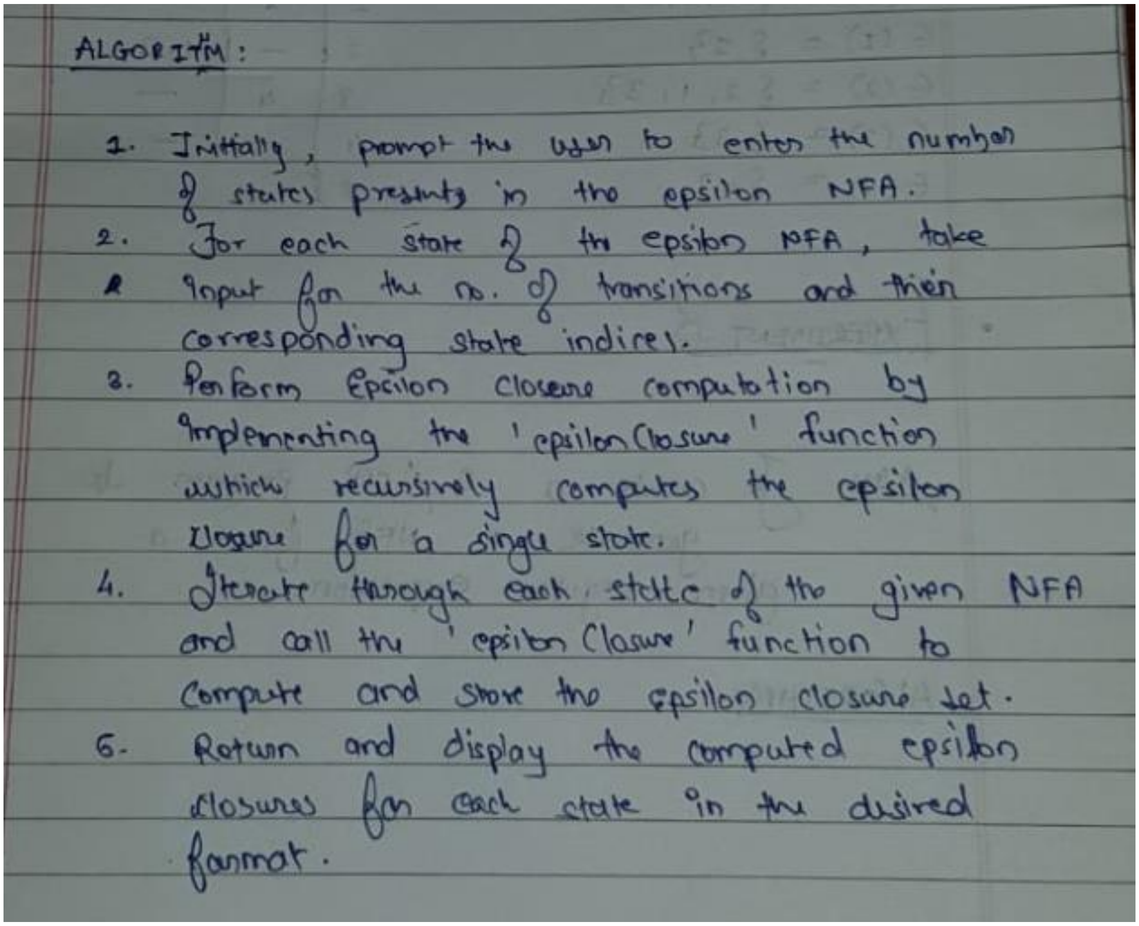
Name: Ojas Patil
Reg No: 21BAI1106

AIM



AIM: To create a C/CPP program to compute the ϵ-closure of all the states of an ϵ-NFA.

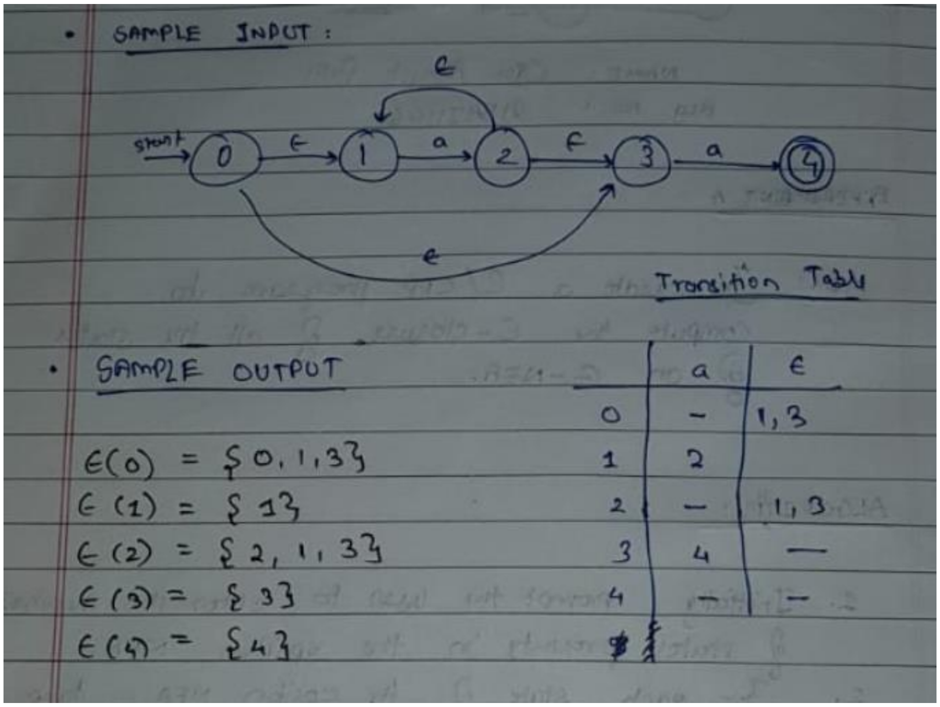
ALGORITHM



ALGORITHM:

1. Initially, prompt the user to enter the number of states presents in the epsilon NFA.
2. For each state of the epsilon NFA, take input for the no. of transitions and their corresponding state indices.
3. Perform Epsilon Closure computation by implementing the 'epsilonClosure' function which recursively computes the epsilon closure for a single state.
4. Iterate through each state of the given NFA and call the 'epsilon Closure' function to compute and store the epsilon closure set.
6. Return and display the computed epsilon closures for each state in the desired format.

Sample Input and Output



- SAMPLE INPUT :

Transition Table

	a	ϵ
0	–	1,3
1	2	
2	–	1,3
3	4	—
4	–	–

- SAMPLE OUTPUT

$\epsilon(0) = \{0,1,3\}$

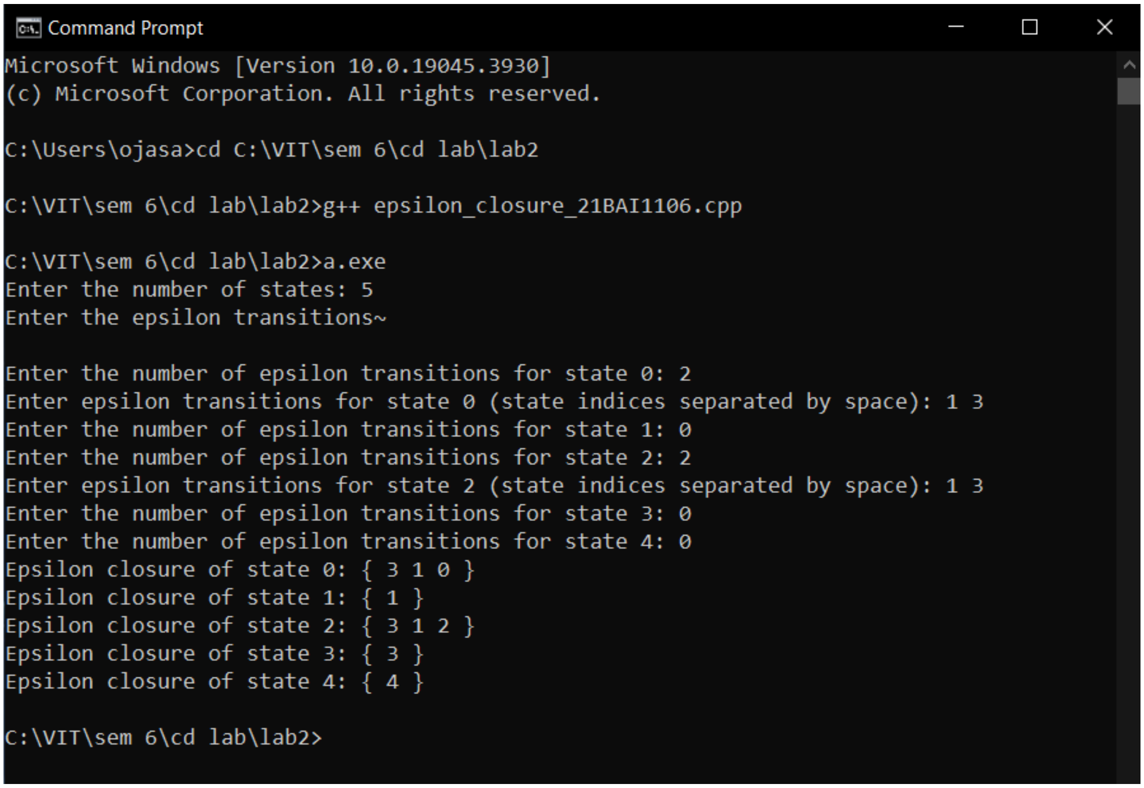
$\epsilon(1) = \{1\}$

$\epsilon(2) = \{2,1,3\}$

$\epsilon(3) = \{3\}$

$\epsilon(4) = \{4\}$

Output Snapshot



```
Microsoft Windows [Version 10.0.19045.3930]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ojasa>cd C:\VIT\sem 6\cd lab\lab2

C:\VIT\sem 6\cd lab\lab2>g++ epsilon_closure_21BAI1106.cpp

C:\VIT\sem 6\cd lab\lab2>a.exe
Enter the number of states: 5
Enter the epsilon transitions~

Enter the number of epsilon transitions for state 0: 2
Enter epsilon transitions for state 0 (state indices separated by space): 1 3
Enter the number of epsilon transitions for state 1: 0
Enter the number of epsilon transitions for state 2: 2
Enter epsilon transitions for state 2 (state indices separated by space): 1 3
Enter the number of epsilon transitions for state 3: 0
Enter the number of epsilon transitions for state 4: 0
Epsilon closure of state 0: { 3 1 0 }
Epsilon closure of state 1: { 1 }
Epsilon closure of state 2: { 3 1 2 }
Epsilon closure of state 3: { 3 }
Epsilon closure of state 4: { 4 }

C:\VIT\sem 6\cd lab\lab2>
```

Program Explanation

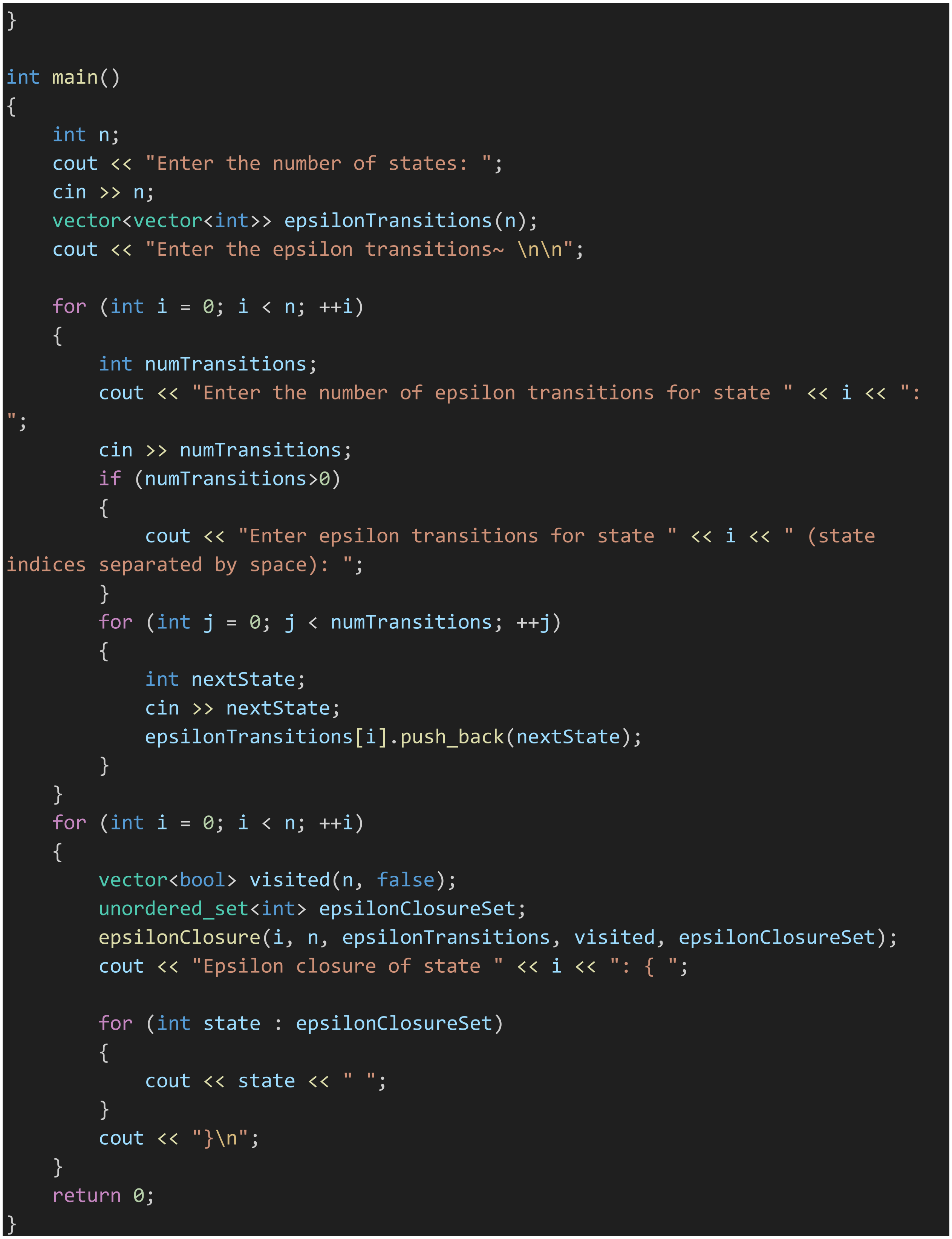
This is a CPP Program which performs the computation of Epsilon Closure of all states of a NFA given its transition table as input. Here is a 6 point breakdown of its explanation:

1. **Function Definition of epsilonClosure:** Defines a function to calculate the epsilon closure of a state in a nondeterministic finite automata (NFA).
2. **Epsilon Closure Computation**: Recursively calculates the epsilon closure of a state by traversing epsilon transitions, marking visited states to avoid infinite loops.
3. **Main Function**: Initializes variables and vectors for epsilon closure computation, prompts user for the number of states (n) and epsilon transitions.
4. **Epsilon Transition Input**: For each state, user inputs the number of epsilon transitions and the reachable state indices via epsilon transitions, constructing epsilonTransitions vector.
5. **Epsilon Closure Computation** (Main Loop): Iterates over each state, computes its epsilon closure using epsilonClosure, and prints the result in the format "Epsilon closure of state i: { set }."
6. **Output**: Outputs epsilon closure sets for all states in the NFA, indicating states reachable through epsilon transitions from each state.

Source Code

```cpp
#include <iostream>
#include <vector>
#include <unordered_set>
using namespace std;

void epsilonClosure(int state, int n, vector<vector<int>> &epsilonTransitions,
vector<bool> &visited, unordered_set<int> &epsilonClosureSet)
{
    visited[state] = true;
    epsilonClosureSet.insert(state);
    for (int i = 0; i < epsilonTransitions[state].size(); ++i)
    {
        int nextState = epsilonTransitions[state][i];
        if (!visited[nextState])
        {
            epsilonClosure(nextState, n, epsilonTransitions, visited,
epsilonClosureSet);
        }
    }
```



```
}

int main()
{
    int n;
    cout << "Enter the number of states: ";
    cin >> n;
    vector<vector<int>> epsilonTransitions(n);
    cout << "Enter the epsilon transitions~ \n\n";

    for (int i = 0; i < n; ++i)
    {
        int numTransitions;
        cout << "Enter the number of epsilon transitions for state " << i << ": ";
        cin >> numTransitions;
        if (numTransitions>0)
        {
            cout << "Enter epsilon transitions for state " << i << " (state indices separated by space): ";
        }
        for (int j = 0; j < numTransitions; ++j)
        {
            int nextState;
            cin >> nextState;
            epsilonTransitions[i].push_back(nextState);
        }
    }
    for (int i = 0; i < n; ++i)
    {
        vector<bool> visited(n, false);
        unordered_set<int> epsilonClosureSet;
        epsilonClosure(i, n, epsilonTransitions, visited, epsilonClosureSet);
        cout << "Epsilon closure of state " << i << ": { ";

        for (int state : epsilonClosureSet)
        {
            cout << state << " ";
        }
        cout << "}\n";
    }
    return 0;
}
```