Batch G1 and G2

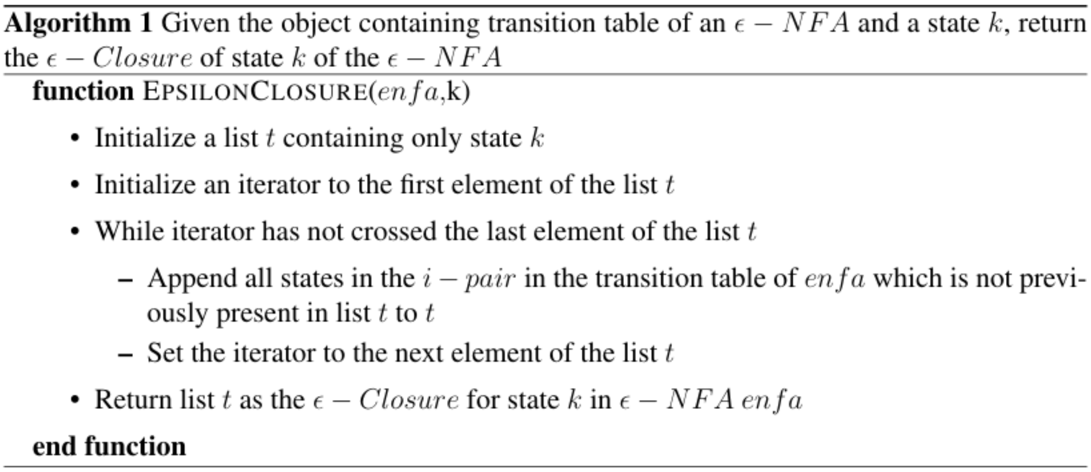
Dated 13-08-2025

Program 1: Implementation of Lexical Analyzer using Lex Tool. In this given program, you are tokenizing a custom language using lex tool. In output, it must give regular expressions and their corresponding return value.

```
Prog
        Integer a, b
        Begin
                read n;
                if a < 10
                then
                    b :=1;
                    else;
                endif
                while a < 10
                do
                      b := 5*a;
                      a := a+1;
                endwhile;

                write a;
                write b;
end
```

Program 2: Write a program to find closure of all states of any given NFA with-transitions



Algorithm 1 Given the object containing transition table of an $\epsilon - NFA$ and a state k, return the $\epsilon - Closure$ of state k of the $\epsilon - NFA$

```
function EpsilonClosure($enfa$,k)
    • Initialize a list $t$ containing only state $k$
    • Initialize an iterator to the first element of the list $t$
    • While iterator has not crossed the last element of the list $t$
        – Append all states in the $i - pair$ in the transition table of $enfa$ which is not previously present in list $t$ to $t$
        – Set the iterator to the next element of the list $t$
    • Return list $t$ as the $\epsilon - Closure$ for state $k$ in $\epsilon - NFA\ enfa$
end function
```

Instructions

Due date: 12.08.2025, 11:30 A.M for Batch G1, and 12.08.2025, 1. 30 P.M for Batch G2 (30 minutes extra is given for the documentation)

Cut-off date: 14.08.2025, 08:00 P.M. (Late submission deadline)

Note:

1. If the submission is after the due date, 2 mark is deducted for every day till cut-off date.
2. If the output is not shown for any of the above component the mark cannot be given even if it is included in your documentation.

Program 3: Calculate Min Time

You are given an integer n and a directed graph with n nodes labeled from 0 to n - 1. This is represented by a 2D array edges, where edges[i] = [ui, vi, starti, endi] indicates an edge from node ui to vi that can only be used at any integer time t such that starti <= t <= endi.

You start at node 0 at time 0.

In one unit of time, you can either:

Wait at your current node without moving, or

Travel along an outgoing edge from your current node if the current time t satisfies starti <= t <= endi.

Return the minimum time required to reach node n - 1. If it is impossible, return -1.

Example 1:

Input: n = 3, edges = [[0,1,0,1],[1,2,2,5]]

Output: 3

Instructions

Due date: 12.08.2025, 11:30 A.M for Batch G1, and 12.08.2025, 1. 30 P.M for Batch G2 (30 minutes extra is given for the documentation)

Cut-off date: 14.08.2025, 08:00 P.M. (Late submission deadline)

Note:

1. If the submission is after the due date, 2 mark is deducted for every day till cut-off date.
2. If the output is not shown for any of the above component the mark cannot be given even if it is included in your documentation.