UCS 802 COMPILER CONSTRUCTION LAB ASSIGNMENT 1

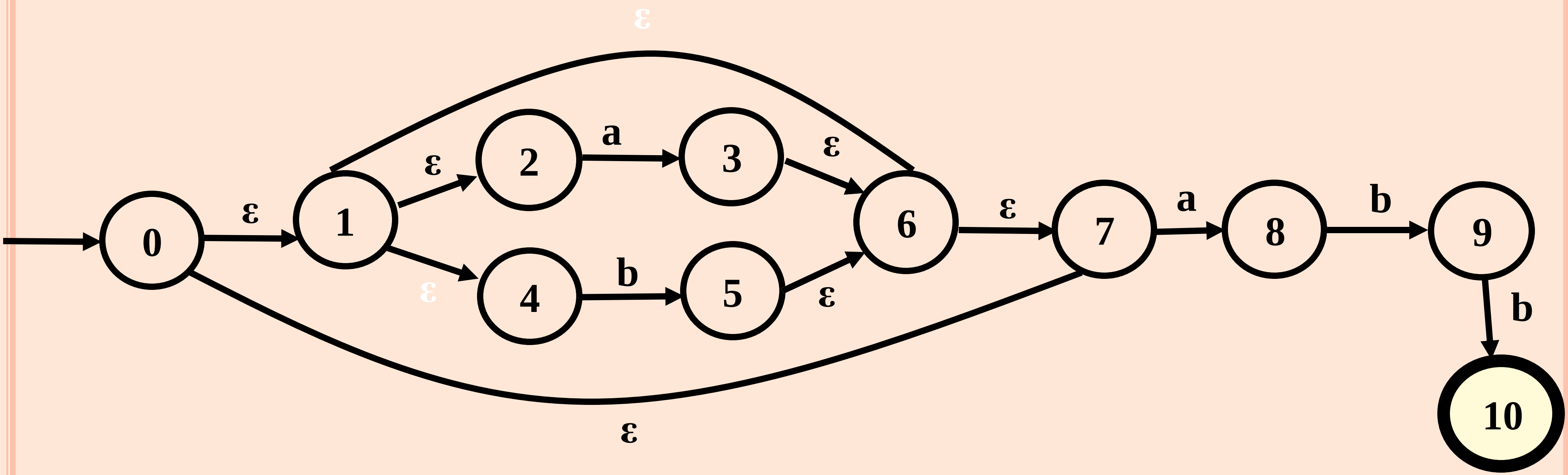
LAB ASSIGNMENT 1

Design a Minimized DFA for the Regular Expression (a/b)*abb i.e. All strings ending with abb.
This will involve three steps:
Generate the NFA using Thomson' s Construction (2 Marks) (1 Lab of 2 Hrs.)
Generate the DFA using Subset Construction (6 marks) (3 Labs of 2 Hrs. each)
Minimize the DFA generated (2 Marks)(2 Labs of 2 Hrs. each)

OR

Design a Minimized DFA for the generic Regular Expression
This will involve three steps:
Generate the NFA using Thomson' s Construction (2 Marks) (1 Lab of 2 Hrs.)
Generate the DFA using Subset Construction (6 marks) (3 Labs of 2 Hrs. each)
Minimize the DFA generated (2 Marks) (2 Labs of 2 Hrs. each)

GIVE THE THOMPSON'S CONSTRUCTION for (a/b)*abb



STEPS TO BE FOLLOWED

- Draw the NFA using Thomson’s on a paper and count the number of states.
- Generate a table of number of states x4, (Why 4: 2 for inputs of Σ, Σ may be 0,1 or a,b and 2 for ε moves on every state; in Thompson’s construction we cannot have more then two out moves from a particular state).
- For RE (a/b)*abb we have a table of 11x 4 since we have 11 states (0 to 10)
- Define the start and final state.

State	a	b	ε_1	ε_2
0	-	-	1	7
1	-	-	2	4
…				
…				
10	-	-	-	-

STEPS TO BE FOLLOWED

1. Find ε –closure (ε –closure will start from state 0)
2. ε -closure(0) will include 0 and contents of ε1 and ε2 corresponding to state 0 in
   the table i.e. 1, and 7; since 1 and 7 are included. Check ε1 and ε2 of 1 and include them, then check ε1 and ε2 of 7 and include them and keep on moving in this pattern (Algorithm on slide No. 6).
3. Marks the set generated as set A.
4. Perform Subset construction (Algorithm on slide No. 7)
5. To identify whether the new set say 'B' generated in the next step is same as previously generated set A , sort all the elements in the new set and compare the length of two sets, if the length of the sets is same, compare the elements else give it a new name.
6. Repeat steps 1-5 for every state having 'a' or 'b' as input

ALGORITHM FOR E-CLOSURE(T)

```
push all states of T onto stack
initialize $\epsilon$-closure(T) to T
while (stack is not empty) do
        begin
        pop t, the top element, off stack;
        for (each state u with an edge from t to u labelled $\epsilon$ do
                begin
                if (u is not in $\epsilon$-closure(T)) do
                        begin
                        add u to $\epsilon$-closure(T)
                        push u onto stack
                        end
                end
        end
```

ALGORITHM FOR CONVERTING A NFA INTO A DFA (SUBSET CONSTRUCTION)

```
put  $\varepsilon$-closure($\{s_0\}$) as an unmarked  state into the set of DFA (DS)
while (there is one unmarked $S_1$ in DS) do
    begin
        mark $S_1$
        for each input symbol a do
          begin
            $S_2$ □   $\varepsilon$-closure(move($S_1$,a))
            if ($S_2$ is not in DS) then
                  add $S_2$ into DS as an unmarked state
            transfunc[$S_1$,a] □   $S_2$
          end
       end
```

ε-closure($\{s_0\}$) is the set of all states can be accessible from s_0 by ε-transition.

set of states to which there is a transition on a from a state s in S_1

- a state S in DS is an accepting state of DFA if a state s in S is an accepting state of NFA
- the start state of DFA is ε-closure($\{s_0\}$

Once we generate the DFA, check whether the DFA is minimized or not.

This can be done by any method discussed in Lec 01 Section 1.5

You can write the program in **any language** of your choice.

You can refer to first 7 videos of
https://www.youtube.com/playlist?list=PLpDGCxH86rUGDvxnONJfICXzIn_eT-5T3

for first assignment of Compiler Construction

DESIRED OUTPUT

Input will be any RE and output should be ‘Accept’ if RE end in ‘abb’ and it should be ‘Not Accepted’ if RE does not end in ‘abb’

TOTAL MARKS: 10