1. The Pseudocode for the Bully Algorithm:

program Bully Algorithm
define
    failed : boolean {true if current leader fails}
    L : process {identifies leader}
    m : message of election, leader, reply
    state : idle, wait_leader, wait_reply

initially state = idle {for every process}

1. do failed -> ∀ j : j > i send election to j;
               state := wait_reply;
               failed := false

2. (state=idle) AND (m=election) -> send reply to sender;
                                   failed := TRUE

3. (state=wait_reply) AND (m=reply) ->
   state := wait for leader

4. (state = wait_reply) AND timeout
   -> L(i) := i;
      ∀ j : j > i :: send leader to j;
      state := idle

5. (state = wait_leader) AND (m = leader)
   -> L(i) := sender;
      state := idle

6. (state = wait_leader) AND (timeout)
   -> failed := true;
      state := idle

od

1.1 Work out the FSM for this algorithm
1.2 Provide thread based implementation of the FSM using thread IPC and buffer pool management module
1.3 Show the execution of the algorithm in the graph below where node 2 finds out leader (node 5) is down. Implement time-outs using UNIX alarm function.

![Graph showing nodes 1 to 5. Node 2 finds out node 5 is down.](image)

2. Chang Roberts Election Algorithm:

Initially all initiator processes are red.

{For each initiator i}

do   
    token <j> ∧ j < i → skip {j’s token is removed}
        
    token <j> ∧ j > i → send token <j>; color := black {i resigns}
        
    token <j> ∧ j = i → L(i) := i {i becomes the leader}

od

{Non-initiators remain black, and act as routers}

{for a non-initiator process}

do   
    token <j> received → color := black; send <j>

od

2.1 Work out the FSM for this algorithm

2.2 Provide thread based implementation of the FSM using thread IPC and buffer pool management module

2.3 Show the execution of the algorithm in a ring with 8 processes as 2,1,5,8,4,6,3,7 order and nodes 1 and 6 start election concurrently.

Your report should describe the FSMs for both algorithms

Grading:  
Demo 40%
Code Analysis 50%
Report 10%