Q1. (15 points) Here is a description of Dekker's algorithm, the first known solution to the mutual exclusion problem for two processes.

```
program dekker (for two processes 0 and 1)
define flag[0], flag[1] : shared boolean;
    turn: shared integer
initially flag[0] = false, flag[1] = false, turn = 1

{program for process 0}
do true []
    flag[0] = true;
do (flag[1] []
    if turn = 1 then
        flag[0] := false;
do turn = 1 [] skip od;
    flag[0] := true;
    fi;
do
    critical section;
    flag[0] = false; turn := 1;
    non-critical section codes;
do
```
{program for process 1}

```plaintext
do  true []
  flag[1] = true;
do (flag[0] [])
    if turn = 0 then
      flag[1] := false;
      do turn = 0 [] skip od;
    fi;
  od

  critical section;
  flag[1] = false; turn := 0;
  non-critical section codes;

od
```

Study the above algorithm. The answer the following questions:

(a) What are the possible values of the shared variables when process 0 enters its critical section?

(b) If both processes want to enter their critical sections infinitely often, then what is the maximum number of times one process can enter its critical section, before the other process does so? Explain your answer.

Q2. (20 points) Consider solving the distributed snapshot problem using a mobile agent on a network whose topology is a strongly connected graph. The initiator sends out a mobile agent with a briefcase that is its data storage. The briefcase has a variable S, which is an array of local states of the various processes. Initially, when the agent is launched, []: S[i] = []. At the end, the agent has to return to the initiator, and S will represent a consistent global state.

Q3. (15 points) In Chang-Roberts algorithm for spanning tree generation, only one designated process can be the initiator. For the sake of speed-up, consider a modification in which any process can initiate the algorithm. So it is quite possible that there can be more than one initiator nodes. Assume that there are two initiators. Suggest how you can construct a spanning tree. Explain your strategy and you algorithm. Does it lead to speed-up? Justify your answer.