Southern Methodist University
Bobby B. Lyle School of Engineering
Department of Computer Science
CS 7343/5343 Operating Systems and System Software

- Include a front page with course title, your name, Student ID, your e-mail address, and the course number (e.g. CS 5343 or CS 7343). You must also indicate whether you are a distance student or an in-person student.
- Each answer should begin in a new page.
- **Note**:
  - There will be 10 points deduction if this information is missing.
  - Late Homework submission must be sent directly to the grader via email.

  - **All students who are signed for this course at the CS 7343 must answer all questions.**
  - **All students who are signed for this course at the CS 5343 must answer exactly four questions.**

1. Consider the following program:

```
const int n = 50;
int tally;
void total()
{
   int count;
   for (count = 1; count <= n; count++){
      tally++;
   }
}
```

```
void main()
{
   tally = 0;
   parbegin (total (), total ());
   write (tally);
}
```

The key word **Parabegin** indicates that both calls to the function total are executed concurrently. Determine the proper lower bound and upper bound on the final value of the shared variable **tally** output by this concurrent program. Assume processes can execute at any relative speed and that a value can only be incremented after it has been loaded into a register by a separate machine instruction. You must explain how you've arrived to the final answer.

2. Examine the following pseudo code in which p and q, defined as shown below. A, B, C, D, and E atomic (indivisible) statements. Assume that the main program execute these two processes concurrently.

```
void p()
{
    A;
    B;
    C;
}
```

```
void q()
{
    D;
    E;
}
```

```
Void main()
  {
    Parbegin p(), q() //Call method p() and method q()  simultaneously
  }
```

   a. Show all possible execution paths of this program. For example, a possible execution path would be ABCDE.
   b. Show al impossible that paths of this program. For example, an impossible path would be EDABC.

3. For each of the following thread state transitions, say whether the transition is legal *and* how the transition occurs or why it cannot.

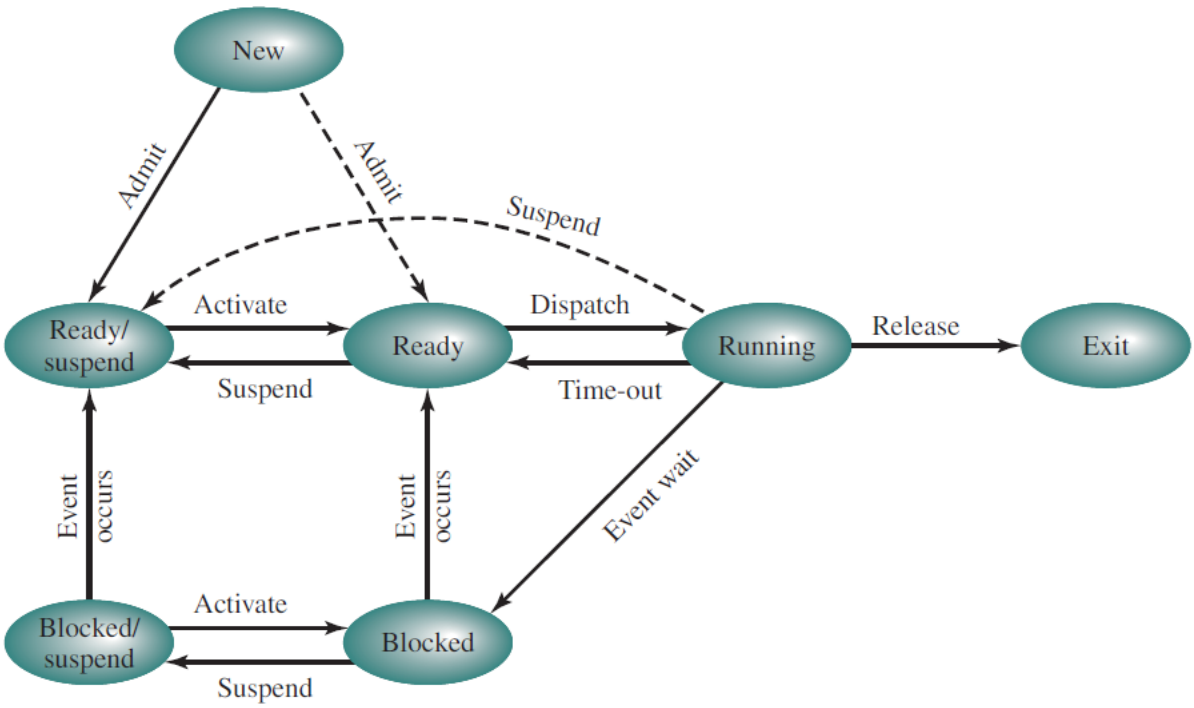
   a. Change from thread state BLOCKED to thread state RUNNING

   b. Change from thread state RUNNING to thread state BLOCKED

   c. Change from thread state RUNNABLE (i.e., in the ready queue) to thread state BLOCKED

4. Consider an environment in which there is an equivalence mapping between user-level and kernel-level threads (i.e. that is we have a one-to-one mapping between user threads and kernel threads) that allows one or more threads within a process to issue blocking system calls while other threads continue to run. Will this approach make multithreaded programs run faster than their single-threaded counterparts on a uniprocessor computer?

5. Consider the figure below. Show all possible transitions and give a scenario in which each transition could occur.



6. Consider the Shell sort algorithm as explained in https://www.youtube.com/watch?v=ddeLSDsYVp8 Is it possible to use two or more threads to implement this algorithm. Explain why/why not.

7. Suppose a process P spawns one or more threads. If the process P terminates and exits, will these threads continue to run? Explain.