Southern Methodist University
Bobby B. Lyle School of Engineering
Department of Computer Science

CS 5343/7343

1. (must be answered by CS 7343 students only)

   The Fibonacci sequence is the series of numbers 0, 1, 1, 2, 3, 5, 8, ... Formally, it can be expressed as:

   $$fib_0 = 0$$
   $$fib_1 = 1$$
   $$fib_n = fib_{n-1} + fib_{n-2}$$

   Write a multithreaded program that generates the Fibonacci sequence. This program should work as follows: On the command line, the user will enter the number of Fibonacci numbers that the program is to generate. The program will then create a separate thread that will generate the Fibonacci numbers, placing the sequence in data that can be shared by the threads (an array is probably the most convenient data structure). When the thread finishes execution, the parent thread will output the sequence generated by the child thread. Because the parent thread cannot begin outputting the Fibonacci sequence until the child thread finishes, the parent thread will have to wait for the child thread to finish. Use the techniques described in Section 4.4 to meet this requirement.

2. This question involves implementing several different process scheduling algorithms. The scheduler will be assigned a predefined set of tasks and will schedule the tasks based on the selected scheduling algorithm. Each task is assigned a priority and CPU burst. Use schedule.txt file to run each algorithm and obtain the ouptout.

   The following scheduling algorithms will be implemented:

   a. First-come, first-served (FCFS), which schedules tasks in the order in which they request the CPU.
   b. Shortest-job-first (SJF), which schedules tasks in order of the length of the tasks' next CPU burst.
   c. Priority scheduling, which schedules tasks based on priority.
   d. Round-robin (RR) scheduling, where each task is run for a time quantum (or for the remainder of its CPU burst).

e. Priority with round-robin, which schedules tasks in order of priority and uses round-robin scheduling for tasks with equal priority.

I. Implementation

The implementation of this project may be completed in either C or Java, and program files supporting both of these languages are provided in the source code download for the text. These supporting files read in the schedule of tasks, insert the tasks into a list, and invoke the scheduler.

The schedule of tasks has the form [*task name*] [*priority*] [*CPU burst*], with the following example format:

```
T1, 4, 20
T2, 2, 25
T3, 3, 25
T4, 3, 15
T5, 10, 10
```

Thus, task `T1` has priority 4 and a CPU burst of 20 milliseconds, and so forth. It is assumed that all tasks arrive at the same time, so your scheduler algorithms do not have to support higher-priority processes preempting processes with lower priorities.