Module 4 (Lectures 16-20) Processes

1. Consider the following pseudo-code program

   ```
   for (i = 0; i < 3; i++)
       fork( );
   ```

   How many processes will be created when this program is executed? Write a C program implementing this functionality and experimentally confirm your answer.
2. A computer system is initially idle. Subsequently, processes are created with the properties shown in the table below. Assume that time starts at 0 and that a process arriving at time *t* does so just before the currently running process (if any) is preempted. The last column indicates the amount of running time that the process requires.

Process Id	Time of creation	Number of Time units of execution required
P1	0	12
P2	3	8
P3	5	6
P4	7	6

   Sketch a time line showing how the execution of these processes will be scheduled by an operating system using Round Robin scheduling with a CPU quantum of 1 time unit.
3. Repeat question 7 assuming instead that the operating system uses the Shortest Job Next non-preemptive scheduling policy. Calculate the process response time (time of process completion minus time of process creation) for each of the 4 processes. Compare the average response time with that under the Round Robin scheduling policy of question 7.
4. Repeat question 7 assuming that the CPU quantum is 4 time units instead of 1 time unit. Once again, calculate the average response time.
5. Linux provides a mechanism called *nice* using which you can change the scheduling priority of a process. Read the manual entry for *nice*. Experiment with it while running programs on Linux. Is it useful in reducing program execution time?