CSE 4300 Homework 2 (Due on October 30th, 2019)

Question 1 (12 points): Many CPU-scheduling algorithms are parameterized. For example, the RR algorithm requires a parameter to indicate the time slice. Multilevel feedback queues require parameters to define the number of queues, the scheduling algorithm for each queue, the criteria used to move processes between queues, and so on.

These algorithms are thus really sets of algorithms (for example, the set of RR algorithms for all time slices, and so on). One set of algorithms may include another (for example, the FCFS algorithm is the RR algorithm with an infinite time quantum). What (if any) relation holds between the following pairs of algorithm sets?

a. Priority and SJF
b. Multilevel feedback queues and FCFS
c. Priority and FCFS
d. RR and SJF

Question 2 (25 points): Calculate the completion time and wait time of the jobs in the following table for FCFS, RR (quantum = 2) and SRTF (shortest remaining time first). The scheduler can break ties arbitrarily.

Job	Length	Arrival time	Completion time			Wait time		
			FCFS	RR	SRTF	FCFS	RR	SRTF
1	50	0						
2	40	5						
3	30	10						
4	20	15						

Question 3.1 (5 points): In real-time scheduling theory, what is the difference among periodic task, sporadic task and aperiodic task?

Question 3.2 (15 points): Given a synchronous task set of three periodic tasks: T1 = (1, 6, 6), T2 = (2, 8, 5) and T3 = (3, 12, 12), please construct the schedules for the task set from time 0 to time 24 under:

(1) Rate-Monotonic Scheduling (RM)
(2) Deadline-Monotonic Scheduling (DM)

(3) Earliest-Deadline-First Scheduling (EDF)

Question 3.3 (5 points): Calculate the average response time of the above three constructed schedules.

Question 4 (8 points): Please explain what is the priority inversion problem. Can priority inversion problem happen if you use round-robin scheduling instead of priority scheduling?

Question 5 (10 points): Using the following example, what is the **exact output** from both the parent process and the child process?

```
#include <unistd.h>
#include <stdio.h>
int main(){
   int i=0;
   i+=1;
   if(fork()){
      i+=2;
      printf("A %d\n",i);
   }else{
      i-=2;
      printf("B %d\n",i);
   }
   return 0;
}
```

Question 6 (20 points): The first known correct software solution to the critical-section problem for *n* processes with a lower bound on waiting of *n* − 1 turns was presented by Eisenberg and McGuire. The processes share the following variables:

```
enum pstate {idle, want_in, in_cs};
pstate flag[n];
int turn;
```

All the elements of flag are initially idle. The initial value of turn is immaterial (between 0 and n-1). The structure of process *Pi* is shown as follows. Prove that the algorithm satisfies all three requirements for the critical-section problem.

```
do {
    while (true) {
        flag[i] = want_in;
        j = turn;

        while (j != i) {
            if (flag[j] != idle)
                j = turn;
            else
                j = (j + 1) % n;
        }

        flag[i] = in_cs;
        j = 0;
        while ((j < n) && (j == i || flag[j] != in_cs))
            j++;

        if ((j >= n) && (turn == i || flag[turn] == idle))
            break;
    }

    /* critical section */

    j = (turn + 1) % n;

    while (flag[j] == idle)
        j = (j + 1) % n;

    turn = j;
    flag[i] = idle;

    /* remainder section */
} while (true);
```