# CSE4300 Homework 1 (Due on Mar 13th, 2019)

**Q1.** What is the purpose of CPUs providing two modes of operation? Give a possible use of these two modes. **(10 points)**

**Q2.** What are the three major activities of memory management in an operating system? **(10 points)**

**Q3.** What is the purpose of system calls? **(10 points)**

**Q4.** What is a microkernel structure? **(10 points)**

**Q5.** Including the initial parent process, how many processes are created by the program shown in the following program? **(15 points)**

```
#include <stdio.h>
#include <unistd.h>

int main()
{
     int i;
     for (i = 0; i < 10; i++)
          fork();
     return 0;
}
```

**Q6.** Explain the role of the init process on UNIX and Linux systems in regard to process termination. **(10 points)**

**Q7.** Explain why interrupts are not appropriate for implementing synchronization primitives in multiprocessor systems. **(10 points)**

**Q8.** 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. **(25 points)**

```
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);
```