CIS 520 – Quiz #1 – Fall 2010 Name: _____

Date: Wed., Oct. 6, 2010
Total: 100 pts

1. (20 points) When programming scientific code with multiple threads, sometimes it is useful to have all threads rendezvous at a place in the code. This is normally done with an operation called a "barrier". It works so that when threads call the Barrier() function, none return until all threads have called the function. For example you might find code that looks like:

```
{
    ThreadsRunInParallel();
    // Threads may reach the barrier at different times

    Barrier(ThreadID());

    // Threads should start here together after the barrier
    ThreadsDoSomethingElseInParallel();
}
```

Your job is to write the **`Barrier()`** function using only semaphores. You are not permitted to use shared variables other than the semaphores. Be sure to show your initial values for the semaphore. You can assume there is a global constant NUM_THREADS that indicates the number of threads in the system. Thread IDs are integers starting at 0 and going to NUM_THREADS-1. NUM_THREADS is less than 100. To simplify things you can assume that **`Barrier()`** is only called once and all threads participate in the barrier.

Semaphore declaration and initialization:

```
void Barrier(int threadID) {




}
```

2. (20 points) In this question you are asked to compare static and dynamic linking.

   a. Describe the advantages of dynamic linking over static linking.

   b. Describe the problems in dynamic linking that are not present in static linking.

   c. If a set of library functions are typically used by at most one process in execution, would it make more sense to use static or dynamic linking to link to the functions? Explain briefly.

   d. If a set of library function are typically used by many processes, would it make more sense to use static or dynamic linking? Explain briefly.

3. (20 points) Deadlock questions:

   a. If you have a system in which there is circularity in the wait-for graph of the system. Is this a definitive sign of deadlock? _____ Explain briefly.

   b. Upon execution, can the following program deadlock? _____.

```
struct semaphore S;
sema_init(&S, 4); // suppose S represents a resource with 4 units

void resource_user_thread(void *_) {
  while (1) {
    // get two units of S
    sema_down(&S);
    sema_down(&S);
    // (use resource)
    // return two units of S
    sema_up(&S);
    sema_up(&S);
  }
}
...
tid_t t1 = thread_create(..., resource_user_thread, NULL);
tid_t t2 = thread_create(..., resource_user_thread, NULL);
tid_t t3 = thread_create(..., resource_user_thread, NULL);
```

   c. If so, give a specific execution sequence that leads to deadlock. If not, explain briefly why deadlock is not possible.

4. Semaphore questions. Suppose you have 2 semaphores and 3 concurrent threads as shown in the example below. Assume that the threads run for a sufficiently long time to generate all possible output using a preemptive scheduler.

```
struct semaphore S, U;
sema_init(&S, 2);
sema_init(&U, 0);
```

```
// Thread 1
while (1) {
  sema_down(&S);
  putchar('A');
  sema_up(&U);
}
```

```
// Thread 2
while (1) {
  sema_down(&U);
  putchar('B');
  putchar('C');
  sema_up(&U);
}
```

```
// Thread 3
while (1) {
  sema_down(&U);
  putchar('D');
}
```

a. How many times will 'A' be printed?

b. What is the minimum number of 'B's that will be printed?

c. How many 'D's will be printed?

d. Can the output start with a 'D'? Explain briefly.

5. (20 points) Static priority scheduling: Assume you have a system with a static priority CPU scheduler. Assume that the scheduler supports preemption but not priority donation. Assume that the priorities are set such that thread T2 has a high priority, thread T1 has the middle priority, and thread T0 has the low priority. Assume the system starts with only thread T0 executing T0( ). Assume that the semaphore `mutex` is initialized to 1.

```
void T0() {
 printf("T0-S\n");

 StartThread(T1);

 printf("T0-E\n");

}
```

```
void T1() {
 printf("T1-S\n");

 P(mutex);

 printf("T1-1\n");

 StartThread(T2);

 printf("T1-2\n");

 V(mutex);

 printf("T1-E\n");
}
```

```
void T2() {

 printf("T2-S\n");

 P(mutex);

 printf(T2-1\n");

 V(mutex);

 printf("T2-E\n");

}
```

a. What is ouput when the code is executed?

b. What changes, if any, would occur in the output if priority donation was added to the scheduler?

c. Can the threads deadlock? Explain briefly.