# CprE 308 Quiz 2

Department of Electrical and Computer Engineering
Iowa State University

You can write on the back of the sheet.

Student Name:

*1.* (3 points) Please answer the following Yes/No questions:

a) The *pthread_join(thread_t tid, void **status)* function blocks the calling thread until the thread specified by *tid* terminates. The specified thread must be in the current process.

YES

*b) With pthread_mutexes, only the thread which currently holds the lock can unlock it.*

YES

*c)* Disabling Interrupts during critical section can be used in the mutual exclusion solutions on both single-processor and multi-processor systems.

NO

2. (3 points) Please answer the following questions:
   a) What if we changed the order of lock() and down() in the previous slide?
   b) What if we changed the order of the unlock() and up()?

**PRODUCER:**

```
While (TRUE) {
        item = produce();

      down (Empty);

        lock(mutex);
      insert(item, buffer);
      count++;
      unlock(mutex);
        up(Full);
}
```

**CONSUMER:**

```
While (TRUE) {
        down (Full);
        lock (mutex);
        item =
        remove(buffer);
        count --;
        unlock(mutex);
        up(Empty);
        consume(item);
}
```

Answer:

a) Likely cause deadlock.

b) No problem.

3. (3 points) Write a program using two threads that can always produce “Iowa State University”.
   – Thread 1 prints “Iowa State”

- Thread 2 prints “University”

Hint: Use condition variable.

```
int    thread1_done = 0;
pthread_cond_t    cv;
pthread_mutex_t  mutex;
```

Thread 1:

```
printf(“Iowa State “);

pthread_mutex_lock(mutex);
thread1_done = 1;
pthread_cond_signal(cv);
pthread_mutex_unlock(mutex);
```

Thread 2:

```
pthread_mutex_lock(mutex);

while (thread1_done == 0) {
        pthread_cond_wait(cv, mutex);
}

printf(“ University\n“);
pthread_mutex_unlock(mutex);
```

main()

```
//init variables, mutexes, cv
//create threads
//join threads
```