On my honor, I have not given, nor received, nor witnessed any unauthorized assistance on this work.

Print name and sign: ______________________________

| Question: | 1 | 2 | 3 | 4 | Total |
|---|---|---|---|---|---|
| Points: | 5 | 8 | 5 | 12 | 30 |
| Score: | | | | | |

1. (5 points) Could the following scenario lead to deadlock? If it could, draw a diagram showing the deadlock.

   Thread A will try to grab first lock 1 and then lock 2. Thread B will try to grab first lock 2 and then lock 3. Thread C will try to grab first lock 1 and then lock 3.

2. (8 points) In order for deadlock to occur, four conditions (listed below) must be met. Give a brief explanation of why each is needed for deadlock to occur.

   (a) Mutual exclusion

   (b) Hold-and-wait

   (c) No preemption

   (d) Circular wait

3. (5 points) Explain the difference between a **mutex** and **semaphore**.

4. Dr. Summet is working on her implementation of locks for some multi-threaded code. Here's what she's got so far (minus header and `include` code):

```
int lock = 0;   //0 for unlocked, 1 for locked
int shared = 0; //shared variable

void* incrementer(void* args){
  for(int i = 0; i < 100; i++){
    //check lock
    while(lock > 0) {}  //spin until unlocked

    lock = 1; //set lock
    shared++; //increment
    lock = 0; //unlock
  }
  return NULL;
}

int main(int argc, char * argv[]){
    int num_threads;
    // Declare an array of threads
    pthread_t threads[num_threads];

    // Create those threads!
    long i;
    for (i = 0; i < num_threads; i++) {
        pthread_create(&threads[i], NULL, print_thread_id, (void *) i);
    }
    // Use pthread_join to make the main thread pause
    for (i = 0; i < NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }
    printf("Shared value: %d; should be 200\n", shared);
    return 0;
}
```

However, no matter how many times she runs her code, she's still getting weird results:

```
    Shared value: 198; should be 200
    Shared value: 208; should be 200
```

(a) (5 points) Explain why Dr. Summet's solution is not atomic and thus doesn't work.

(b) (3 points) You try to explain to Dr. Summet how she should fix her code. Should she use a semaphore or a mutex? Why?

(c) (4 points) Correct her code on the previous page using the synchronization primitive you selected in the previous part. You do not need to have syntactically correct code, but you should demonstrate you understand how to guarantee atomicity.