CS162 Section 1 (9/6/12)

True/False

1. Threads within the same process share the same heap and stack.

2. Preemptive multithreading requires threads to give up the CPU using the yield() system call.

3. Despite the overhead of context switching, multithreading can provide speed-up even on a single-core cpu.

Short Answers

1. What is the OS data structure that represents a running process?

2. What are some of the similarities and differences between interrupts and system calls? What roles do they play in preemptive and non-preemptive multithreading?

Concurrency Problem

Threads A and B both share references to the same newly initialized List object. Each thread calls the add() method once. Consider the three scenarios below where thread A runs initially and then context switches to thread B which completes the entire add() method before switching back to thread A. Which of these scenarios would lead to incorrect behavior of add()? Incorrect behavior means that the final List object has missing Nodes or incorrect pointers.

```
class Node {
   Object entry;
   Node next;
   Node(Object entry) {
      this.entry = entry;
   }
}
```

```
class List {
   Node head = new Node(null);
   Node tail = head;

   void add(Object entry) {
      Node newNode = new Node(entry);    // scenario 1:  thread A contexts switches to B after this line
      Node oldTail = tail;     // scenario 2:  thread A contexts switches to B after this line
      tail = newNode;          // scenario 3:  thread A contexts switches to B after this line
      oldTail.next = newNode;
   }
}
```