CS 162 Section 1 Fall 2013

True/False

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

2. An individual process believes that it has its own address space. **T**

3. Operating systems can have multiple address spaces and can support multiple threads per address space. **T**

4. A process is equivalent to a program. **F**

Short Answer

1. What is the OS data structure that represents a running process?
   Process control block (PCB)

2. What needs to be done when you switch from one virtual "CPU" to another?
   Context switch:
   Save PC, SP, and registers in current state block.
   Load PC, SP, and registers from new state block.

3. What are the two main challenges of multiprogramming, and what is the difference between them?
   Concurrency: Need to arbitrate access to shared resources
   Protection: Need to protect applications from each other

Longer Answer

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: A switches to B after this line
                Node oldTail = tail;                    //scenario 2: A switches to B after this line
                tail  = newNode;                        // scenario 3: A switches to B after this line
                oldTail.next = newNode;
        }
}
```

Scenario 1: Correct behavior. Result is the same as if thread B and then thread A called add() serially.

Scenario 2: Incorrect behavior. Result is as if thread B's add() never occurred.

Scenario 3: Correct behavior. Result is the same as if thread A and then thread B called add() serially.