Q3

Source: Lecture 2 slide 49.

```
boolean running = true;
Thread t1 = new Thread(() → {
      while (running) {
                  /* do nothing */
      }
      System.out.println(“t1 finishing
execution");
})
t1.start();
try{Thread.sleep(500);}catch(…){…}
running = false;
System.out.println("Main finishing
execution");
```

Q3

Source: Lecture 2 slide 58.

```
// shared variables
int x=0; int y=0;
int a=0; int b=0;4½
Thread one = new Thread(() → {
    a=1;
    x=b;
});
Thread other = new Thread(() → {
    b=1;
    y=a;
});
one.start();other.start();
one.join();other.join();
System.out.println("("+x+","+y+")");
```

Q8

Source: Lecture 5 slide 34.

```
1  public void push(T value) {
2      Node newHead = new Node(value);
3      Node oldHead;
4      do {
5              oldHead = top.get();
6              newsHead.next = oldHead;
7      } while (!
8  top.compareAndSet(oldHead,newHead));
9  }
```

```
10 public T pop() {
11     Node newHead;
12     Node oldHead;
13     do {
14     oldHead = top.get();
15     if(oldHead == null) { return null; }
16     newHead = oldHead.next;
17     } while (!
18 top.compareAndSet(oldHead,newHead));
19     return oldHead.value;
20 }
```

Q8

Source: Lecture 5 slide 11.

```
class MyAtomicInteger {
    …
    public int addAndGet(int delta)
{
            int oldValue, newValue;
            do {
                oldValue = get();
                newValue = oldValue+
delta;
            } while (!
compareAndSet(oldValue, newValue));
            return newValue;
    }
public int getAndAdd(int delta) {
            int oldValue, newValue;
            do {
                oldValue = get();
                newValue = oldValue +
delta;
            } while (!compareAndSet(
                        oldValue,
newValue));
            return oldValue;
        }
    …
}
```