Key to CSE143 Midterm, Spring 2021 handout #17

1.

Method Call	Output Produced
mystery(7)	**.7
mystery(246)	*6*4*2*.
mystery(195)	****.195
mystery(1234)	*4**2**.13
mystery(29548)	*8*4***2*.95

2. One possible solution appears below.

```
public int maxDigits(int x, int y) {
    if (x < 0 || y < 0) {
        throw new IllegalArgumentException();
    }
    if (x == 0 && y == 0) {
        return 0;
    } else {
        return 10 * maxDigits(x / 10, y / 10) + Math.max(x % 10, y % 10);
    }
}
```

3.

Statement	Output
var1.method2();	Table 2/Chair 2
var2.method2();	Table 2/Chair 2
var3.method2();	Table 2/Chair 2
var4.method2();	Lamp 2
var5.method2();	Table 2/Chair 2
var6.method2();	compiler error
var1.method1();	Table 1
var2.method1();	compiler error
var3.method1();	Couch 1
var4.method1();	compiler error
var1.method3();	Table 3/Table 1
var2.method3();	compiler error
var3.method3();	Table 3/Couch 1
((Lamp)var4).method1();	Lamp 1
((Lamp)var2).method1();	runtime error
((Table)var5).method1();	Couch 1
((Couch)var1).method1();	runtime error
((Chair)var6).method2();	Chair 2
((Couch)var5).method3();	Table 3/Couch 1
((Table)var5).method3();	Table 3/Couch 1

4.

before	after	code
p->[1] q->[2]->[3]	p->[1]->[3] q->[2]	p.next = q.next; q.next = null;

p->[1]->[2]
 	
q->[3]
	p->[2]
	
	q->[1]->[3]
		ListNode temp = p;
		p = p.next;
		temp.next = q;
		q = temp;

p->[1]->[2]->[3]
 	
q->[4]
	p->[4]->[3]
	
	q->[2]->[1]
		q.next = p.next.next;
		p.next.next = p;
		ListNode temp = q;
		q = p.next;
		p = temp;
		q.next.next = null;

p->[1]->[2]
 	
q->[3]->[4]->[5]
	p->[2]->[3]
	
	q->[5]->[4]->[1]
		p.next.next = q;
		ListNode temp = q;
		q = q.next.next;
		q.next = temp.next;
		q.next.next = p;
		p = p.next;
		p.next.next = null;
		q.next.next.next = null;

5. One possible solution appears below.

```
    public ArrayIntList sublist(int start, int stop) {
        ArrayIntList result = new ArrayIntList();
        for (int i = start; i < stop; i++) {
            result.elementData[i - start] = elementData[i];
        }
        result.size = stop - start;
        return result;
    }
```

6. One possible solution appears below.

```
    public static void alternatingReverse(Stack<Integer> s) {
        if (s.size() % 2 != 0) {
            throw new IllegalArgumentException();
        }

        Queue<Integer> q = new LinkedList<>();
        int oldSize = s.size();
        while (!s.isEmpty()) {
            q.add(s.pop());
        }
        for (int i = 0; i < oldSize / 2; i++) {
            s.push(q.remove());
            q.add(q.remove());
        }
        while (!s.isEmpty()) {
            q.add(q.remove());
            q.add(s.pop());
        }
        while (!q.isEmpty()) {
            s.push(q.remove());
        }
    }
```