

Solution to CSE143X Section #13 Problems

1.	Method Call	Output Produced
	mystery(5);	+
	mystery(15);	++
	mystery(304);	+++
	mystery(9247);	++++
	mystery(43269);	+++++

2. One possible solution appears below.

```
public int doubleDigit(int n, int d) {
    if (d < 0 || d > 9) {
        throw new IllegalArgumentException();
    }
    if (n < 0) {
        return -doubleDigit(-n, d);
    } else if (n == 0) {
        return 0;
    } else if (n % 10 == d) {
        return doubleDigit(n / 10, d) * 100 + d * 11;
    } else {
        return doubleDigit(n / 10, d) * 10 + n % 10;
    }
}
```

3.	Statement	Output
	var1.method2();	Spoon 2
	var2.method2();	Bowl 2
	var3.method2();	Bowl 2
	var4.method2();	Pot 2
	var5.method2();	compiler error
	var6.method2();	Fork 2/Pot 2
	var1.method1();	compiler error
	var2.method1();	Bowl 1
	var3.method1();	compiler error
	var1.method3();	Pot 3/Spoon 2
	var2.method3();	Pot 3/Bowl 2
	var3.method3();	Pot 3/Bowl 2
	var4.method3();	Pot 3/Pot 2
	((Spoon)var1).method1();	Spoon 1
	((Bowl)var3).method1();	Bowl 1
	((Fork)var3).method3();	Pot 3/Bowl 2
	((Fork)var5).method1();	compiler error
	((Spoon)var5).method1();	runtime error
	((Fork)var6).method2();	Fork 2/Pot 2
	((Bowl)var6).method3();	runtime error

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

4.	before	after	code
	p->[1]->[2]	p->[4]->[2]	q.next.next = p.next;
			p.next = q;
	q->[3]->[4]	q->[1]->[3]	q = p;
			p = q.next.next;
			q.next.next = null;
	p->[1]->[2]->[3]	p->[2]->[4]	p.next.next.next = p;
			q.next.next = p.next.next;
	q->[4]->[5]	q->[5]->[3]->[1]	p.next.next = q;
			q = q.next;
			p = p.next;
			p.next.next = null;
			q.next.next.next = null;

5. One possible solution appears below.

```
public ArrayIntList extractOddIndexes() {
    ArrayIntList result = new ArrayIntList();
    for (int i = 0; i < size / 2; i++) {
        result.elementData[i] = elementData[2 * i + 1];
        elementData[i] = elementData[2 * i];
    }
    result.size = size / 2;
    if (size % 2 == 0) {
        size = size / 2;
    } else {
        elementData[size / 2] = elementData[size - 1];
        size = size / 2 + 1;
    }
    return result;
}
```

6. One possible solution appears below.

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