CS 142 Sample Midterm Exam #2

1. Array Mystery

Consider the following method:

```
public static void mystery(int[] list) {
    for (int i = 2; i < list.length; i++) {
        if (i % 2 == 0) {
            list[ i ] = list[ i ] + list[ i - 2 ];
        }
    }
}
```

In the left-hand column below are specific arrays of integers. Indicate in the right-hand column what values would be stored in the array after the call to function mystery in the left-hand column. Write your answer surrounded by curly braces with number separated by commas.

Original Contents of Array	**Final Contents of Array**
`int[] a1 = {2, 5, 8};` `mystery(a1);`	_____
`int[] a2 = {3, 4, 7, 5};` `mystery(a2);`	_____
`int[] a3 = {2, 4, 6, 2, 6};` `mystery(a3);`	_____
`int[] a4 = {4, 5, 8, 9, 3};` `mystery(a4);`	_____
`int[] a5 = {1, 2, 8, 5, 10, 5, 4};` `mystery(a5);`	_____

2. ArrayList Mystery

Write the final contents of each of the `ArrayList`s after it is passed to the following method:

```
public static void mystery5(ArrayList<Integer>list) {
    int size = list.size();
    for (int i = 1; i < size; i++) {
        if(list.get(i) % 2 == 1) {
            list.remove(i);
        } else {
            list.add(i - 1, -1 * list.get(i));
        }
    }
}
```

Your answer should be formatted identically (including spacing) to the inputs listed below.

Original Contents of ArrayList	**Final Contents of ArrayList**
`{34, 2, 5}`	_____
`{12, 34, 65, 1, 4}`	_____
`{54, 23, 7, 2, 4, 2}`	_____

3. Complexity

Give a tight bound of the nearest runtime complexity class for each of the following code fragments in Big-Oh notation, in terms of the variable `N`. In other words, write the code's growth rate as `N` grows. Write a simple expression that gives only a power of `N` using a caret `^` character for exponentiation, such as `O(N^2)` to represent `O(N2)` or `O(log N)` to represent `O(log2 N)`. Do not write an exact calculation of the runtime such as `O(2N3 + 4N + 14)`.

// a)

```
public int[] mystery1(int[] list) {
    int[] result = new int[2 * list.length];
    for (int i = 0; I < list.length; i++) {
        result[2 * i] = list[i] / 2 + list[i] % 2;
        result[2 * i + 1] = list[i] / 2;
    }
    return result;
}
```

// b)

```
public void mystery2(int[] list){
    for(int i = 0; i < list.length / 2; i++){
        int j = list.length - 1 - i;
        int temp = list[i];
        list[i] = list[j];
        list[j] = temp;
    }
}
```

// c)

```
public void mystery3(ArrayList<String> list) {
    for (int i=0; i < list.size()-1; i+=2) {
        String first = list.remove(i);
        list.add(i + 1, first);
    }
}
```

// d)

```
public void mystery4(ArrayList<String> list) {
    for (int i = 0; i < list.size()- 1; i += 2) {
        String first = list.get(i);
        list.set(i, list.get(i + 1));
        list.set(i + 1, first);

        int count = 0;
        while(count < list.size()) {
            count *= 2;
        }
    }
}
```

4. Polymorphism Mystery

Assuming that the following classes have been defined:

```
public class Gulp {
    public void method2() {
        System.out.println("Gulp 2");
        method3();
    }

    public void method3() {
        System.out.println("Gulp 3");
    }
}

public class Sip extends Gulp {
    public void method1() {
        System.out.println("Sip 1");
    }

    public void method3() {
        System.out.println("Sip 3");
    }
}

public class Bite extends Gulp {
    public void method1() {
        System.out.println("Bite 1");
    }

    public void method3() {
        System.out.println("Bite 3");
        super.method3();
    }
}

public class Drink extends Bite {
    public void method3() {
        System.out.println("Drink 3");
    }
}
```

```
var1.method2();   _____
var2.method2();   _____
var3.method2();   _____
var4.method2();   _____
var5.method2();   _____
var6.method2();   _____
var1.method3();   _____
var2.method3();   _____
var3.method3();   _____
var4.method3();   _____
var5.method3();   _____
var6.method3();   _____
((Sip)var6).method1(); _____
((Gulp)var1).method1(); _____
((Gulp)var1).method2(); _____
((Bite)var1).method3(); _____
((Bite)var6).method1(); _____
((Drink)var1).method1(); _____
((Drink)var4).method2(); _____
((Bite)var3).method1(); _____
```

And assuming the following variables have been defined:

```
Object var1 = new Bite();
Gulp var2 = new Gulp();
Gulp var3 = new Sip();
Bite var4 = new Drink();
Object var5 = new Gulp();
Gulp var6 = new Drink();
```

In the table above, indicate in the right-hand column the output produced by the statement in the left-hand column. If the statement produces more than one line of output, indicate the line breaks with slashes as in `a/b/c` to indicate three lines of output with `a` followed by `b` followed by `c`. If the statement causes an error, fill in the right-hand column with either the phrase **`compiler error`** or **`runtime error`** to indicate when the error would be detected.

5. Array Programming

Write a static method called `reverseFirstK` that takes an array of integers and an index *k* as parameters and that and modifies the first `k` elements of the array to be in reverse order.

For example, if a variable called "list" stores the following values:

```
[10, -2, 33, 55, 9, 17, 6]
```

then the call:

```
reverseFirstK(list, 5);
```

should reverse the first 5 elements, modifying it to store:

```
[9, 55, 33, -2, 10, 17, 6]
```

If the value of `k` is 1, 0, negative, or greater than the length of the array, the array should remain unchanged.

You are not allowed to use an auxiliary data structure such as a temporary array or `ArrayList` to solve this problem and you are not allowed to call any methods of the `Arrays` class or the `Collections` class.

6. ArrayList Programming

Write a static method called `kCopies` that accepts an `ArrayList` of integers as a parameter and replaces every integer of value `k` with `k` copies of itself. For example, if a variable called `list` contains the following:

```
[4, 1, 2, 0, 3]
```

and we make the following call:

```
kCopies(list);
```

Then list should store the following values after the call:

```
[4, 4, 4, 4, 1, 2, 2, 3, 3, 3]
```

Zeroes and negative numbers should be removed from the list.

You may not construct any extra data structures to solve this problem. You must solve it by manipulating the `ArrayList` you are passed as a parameter.

7. Critter Programming

Write a critter class `Grasshopper` along with its movement and eating behavior. All unspecified aspects of `Grasshopper` use the default behavior. Write the complete class with any fields, constructors, etc. necessary to implement the behavior.

A `Grasshopper` sits still until getting into a fight. Once it fights, it celebrates its victory by doing a "hop". A "hop" consists of moving north a certain number of times, then south the same number of times, then west one time. The hops start with a height of 1 (one move north, then one move south) but each subsequent fight causes the next hop to be larger by one. The second hop is 2 moves north, 2 moves south, then 1 move west. After finishing the hop, the Grasshopper sits idle again until it gets into another fight.

If a `Grasshopper` is sitting still, it always fights with `Attack.ROAR`. If a `Grasshopper` gets into a fight in the middle of a hop (while it is not sitting still), it always returns `Attack.FORFEIT`, causing it to lose the fight.

Here is an example sequence of moves for one Grasshopper:

`CCCCC` (fights) `NSWCCC` (fights) `NNSSWCCCCCCC` (fights) `NNNSSSWCC` (fights) `NNNNSS` (fights and dies).