Sample Intern Interviews (from 2017 summer interviews)

This exam has 5 questions. You do not need to complete all of the questions--choose the questions you feel the best about. Partial credit is given, so please submit your code even if it is not completely working. Comments are not required, but if your code does not work 100% comments will help for partial credit.

The more difficult questions have more points. The theoretical maximum score is 20, but not expected in the allotted time. The more that you answer the more points you will be given. You do not have to handle error situations, you will only be evaluated on the cases that I give below.

Question 1: String parsing (2 points possible)

Given the following code, modify the **parse** function so that it parses the input string and prints out the three parts of a phone number. Handle all of the cases given below. Rubric: 2 points (2 points for correct output).

```
public class Question1 {
    public static void parse(String str) {
        System.out.println("Area code: " + "");
        System.out.println("Prefix: " + "");
        System.out.println("Number: " + "");
        System.out.println("----------");
    }

    public static void main(String[] args) {
        parse("314-555-1212");
        parse("(636)555-1212");
        parse("636 - 555 - 1212");
        parse("3145551212");
    }
}
```

Question 2: Histogram (4 points possible)

Produce a histogram. Using the input string, count the number of occurrences of each of the characters. Display these counts. Rubric: 4 points (2 point for correct output, 2 point for sorted final output).

```
public class Question2 {
    public static void main(String[] args) {
        String input = "OpMmVlLoLgPsJpLoNkKuGkRnNyKtTlVsPkNkPnPwMoPlWqNuNm"+
                "RjSqVpNlFnVpBjMeNrJtFpOhIoGoLpKsRqLjQmNkHoMeQoQnMo";


    }
}
```

Correct output:

```
B:1
F:2
G:2
H:1
I:1
J:2
```

```
K:3
L:5
M:5
N:8
O:2
P:5
Q:3
R:3
S:1
T:1
V:4
W:1
e:2
g:1
h:1
j:3
k:5
l:4
m:3
n:4
o:8
p:6
q:3
r:1
s:3
t:2
u:2
w:1
y:1
```

Question 3: Sort (4 points possible)

Sort the following numbers and display the results. Rubric: 4 points (2 point for correct output, 2 points creating your own sort algorithm (not using Java's built in sorting)).

```
public class Question3 {
    public void main(String[] args) {
        int[] list = { 6,9,10,8,7,4,3,1,5,2 };
    }
}
```

Question 4: Combinations and Factorial (4 points possible)

I provide you with a factorial function and a combinations (nCr) function. Why do the last two fail, even though the first two are correct? (Answer in a code comment) Implement a better solution that handles all 4. Rubric: 4 points (1 point for description of problem, 3 for a version that handles the last two).

```
public class Question4 {

    public static int factorial(int n) {
        int fact = 1; // this  will be the result
        for (int i = 1; i <= n; i++) {
            fact *= i;
        }
        return fact;
    }

    public static int c(int n, int r) {
        return(factorial(n)/(factorial(r)*factorial(n-r)));
    }

    public static void main(String[] args) {
        System.out.println( "c(6,2) = " + c(6,2)); // Should be 15
        System.out.println( "c(8,4) = " + c(8,4)); // Should be 70
        System.out.println( "c(20,2) = " + c(20,2)); // Should be 190
        System.out.println( "c(10000,1) = " + c(10000,1)); // Should be 10000
    }
}
```

Question 5: Trees (6 points possible)

The following code implements a very basic tree. The tree is loaded and then traversed "in order". Choose a random node from the tree. Do not hard code the length, your random node picker should work for any tree expressed with the code below. Rubric: 6 points (3 choosing a random node, 3 for a random node where all have an equal opportunity (no bias)).

```
public class Question5 {
    public static class Node
    {
        public Node(String name) {
            this.name = name;
        }

        // Add a new node to the tree (or node)
        public void add(String addName) {

            if( addName.compareTo(this.name)>0 ) {
                if( this.right==null ) {
                    this.right = new Node(addName);
                } else {
                    this.right.add(addName);
                }
            } else {
                if( this.left==null ) {
                    this.left = new Node(addName);
                } else {
                    this.left.add(addName);
                }
            }
        }

        public String name;
        public Node left, right;
    }

    // Traverse the tree "in order"
    public static void traverse(Node node) {
        if( node!=null ) {
            traverse(node.left);
            System.out.println(node.name);
            traverse(node.right);
        }
    }

    public static void main(String[] args) {
        Node tree = new Node("lamb");
        tree.add("dog");
        tree.add("zebra");
        tree.add("unicorn");
        tree.add("cat");
        tree.add("lion");

        traverse(tree);

    }
}
```