There will not be this many problems on the exam!!

1. Consider the loop below:

   ```java
   int x[] = {'a', 'b', 'c'};
   for (int i=0; i<x.length; i++)
       System.out.print(x[i]);
   ```

   The code prints `abc`. Rewrite the loop as a while loop. Then rewrite it using the "foreach" construction. In both cases, the output should also be `abc`.

2. The output of the code below is `1 10`. Explain.

   ```java
   public class P2 {
       public static void main(String[] args) {
           int x = 1;
           int y[] = {1, 2};
           f(x, y);
           System.out.println(x + " " + y[0]);
       }

       public static void f(int x, int[] y) {
           x = 10;
           y[0] = 10;
       }
   }
   ```

3. Below are method calls to 3 of the methods of the String class. Which are correct, and which are incorrect? Explain your answers.

   ```java
   String s = "abc";
   char c1 = s.charAt(1);        // (1)
   int len = String.length(s);   // (2)
   String s = String.valueOf(1); // (3)
   ```

4. Consider the code below:

   ```java
   // this is a built-in Java interface, but I've
   // supplied code for your reference
   public interface Comparable<Item> {
       public int compareTo(Item other);
   }

   public class MyInteger implements Comparable<MyInteger> {
       private int x;

       public MyInteger(int x) {
           this.x = x;
   ```
public int compareTo(MyInteger i) {
    return 0;  // replace this
}

public static void main(String[] args) {
    System.out.println("Enter two integers");
    int x = StdIn.readInt();
    int y = StdIn.readInt();
    MyInteger i1 = new MyInteger(x);
    MyInteger i2 = new MyInteger(y);
    System.out.println(i1.compareTo(i2));
}

Fill in the compareTo method such that the following is true:

a. If x < y, then the output is a negative number
b. If x == y, then the output is 0
c. If x > y, then the output is a positive number

5. When is storage space allocated and de-allocated for each of the following types of variables?
   - local variables
   - instance variables
   - static variables (i.e., class variables)

6. Complete the Letters class below. It should be completed in such a way that the program prints a ab abc abcd abcde

```java
import java.util.Iterator;

public class Letters implements Iterable<Integer> {
    // maybe instance variables, maybe not
    private int maximum;

    public Letters(int x) {
        maximum = x;
    }

    // fill this in
    public Iterator<String> iterator() {
        // replace this
        return null;
    }
}
```
public static void main(String[] args) {
    Letters let = new Letters(5);
    for (String s : let)
        System.out.print(s + " ");
    System.out.println();
}

7. Write a main method which does the following in the order specified:
   a) Creates an empty Bag of Strings (the Bag class from lecture)
   b) Adds the Strings "a", "b", and "c" to the bag
   c) Prints the contents of the Bag, without using the Bag's toString method

8. Fill in the intersection method below. The method is passed two Bags, and returns a new Bag which contains all items that are in both of the parameter Bags. Assume that Bag has the methods add, contains, and iterator. Although there may be duplicate elements in the original Bags, write your answer so that the new Bag does not contain any duplicates.

   public class BagUtility<Item> {
       public static Bag<Item> intersection(Bag<Item> bag1, Bag<Item> bag2) {

           // replace this
           return null;
       }
   }

9. Characterize the running time of each of the operations below, based on the amount of data $n$ involved. Specify whether a curve that plots $n$ vs the running time $T(n)$ is a horizontal line, or a line with a positive slope, or perhaps something else.
   i. The contains method
   ii. The remove method
   iii. The intersection method from above

10. In many programming languages such as Java, symbols such as parentheses, brackets, and braces (we'll refer to them all as “parens”) are used to indicate the beginning and end of various parts of a program. Almost all of the time, the parens are balanced, meaning that for each left paren there is a corresponding right paren. Moreover, the left and right parens must occur in a certain ordering; for example, “( ) {}” and “{ [ ] }” have balanced parens, but “{ ( ) }” does not.
Complete the isBalanced method below so that it returns true if the parameter string contains balanced parents, but false if it does not.

```java
public class Balanced {
    // ex1 has balanced parens
    private static String ex1 = "void f() { int i=0; while (i<2) { System.out.println(i); } }";
    // ex2 is missing a right paren
    private static String ex2 = "void f() { int i=0; while (i<2 { System.out.println(i); } }";
    // ex3 is missing a right curly brace at the end
    private static String ex3 = "void f() { int i=0; while (i<2) { System.out.println(i); }";

    public static boolean isBalanced(String code) {
        Stack300<Character> parens = new Stack300<Character>();
        String leftParens = "([{";
        String rightParens = ")]";
        for (int i=0; i<ex1.length(); i++) {
            char next = ex1.charAt(i);
            // if next is one of the left paren symbols, push it on the stack
            int left = leftParens.indexOf(next);
            if (left != -1) {
                parens.push(next);
            } else {
                int right = rightParens.indexOf(next);
                if (right != -1) {
                    parens.pop();
                } else {
                    return false;
                }
            }
        }
        return parens.isEmpty();
    }

    public static void main(String[] args) {
        System.out.println(isBalanced(ex1) + "\n"); // should be true
        System.out.println(isBalanced(ex2) + "\n"); // should be false
        System.out.println(isBalanced(ex3) + "\n"); // should be false
    }
}
```