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

1. Consider the loop below:

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

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`.

**Answers**

```java
int i=0;
while (i < x.length)
    System.out.print(x[i++]);
System.out.println();
```

```java
for (Character c : x)
    System.out.print(c);
System.out.println();
```

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;
    }
}
```

**Answer:** Since Java is pass by value, and since `int` is a primitive type, the local variable `x` in `main` is not modified within `f`. However, since arrays are objects, and since reference to objects are passed as parameters, `y` can be changed within `f`.

3. Below are method calls to 3 of the methods of the `String` class. Which are correct, and which are incorrect? Explain your answers.
String s = "abc";  
char c1 = s.charAt(1);  // (1) **Answer:** correct, since charAt is an instance method
int len = String.length(s);  // (2) **Answer:** incorrect; should be s.length()
String s = String.valueOf(1);  // (3) **Answer:** correct, because valueOf is a class (static) method

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;
    }

    // fill this in
    public int compareTo(MyInteger i) {
        return x - i.x;  // answer
    }
}

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)
**Answer:** storage for local variable is allocated whenever the method in which the variable is declared is called. Storage is de-allocated when the method returns. Storage for instance variables is allocated whenever an object is created which is an instance of the class which the variable is a member of. It is de-allocated if there is no longer anything which refers to (points to) the object. Storage for a class variable is allocated at the beginning of program execution, and is de-allocated when the program terminates.

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<String> {
  // maybe instance variables, maybe not
  private int maximum;

  private String letters = "abcdefghijklmnopqrstuvwxyz";

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

  // fill this in
  public Iterator<String> iterator() {
    return new Iterator<String>() {
      private int counter = 1;

      public Boolean hasNext() {
        return counter <= maximum;
      }

      public String next() {
        return letters.substring(0,counter++);
      }
    };
  }

  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
Answer:

```java
public class P7 {
    public static void main(String[ ] args) {
        Bag<String> b = new Bag<String>();
        b.add("a"); b.add("b"); b.add("c");
        System.out.println(b);  // calls toString implicitly
    }
}
```

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.

Answer:

```java
public class BagUtility<Item> {
    public Bag<Item> intersection(Bag<Item> bag1, Bag<Item> bag2) {
        Bag<Item> ans = new Bag<Item>();
        for (Item i : bag1)
            ans.add(i);
        for (Item i : bag2)
            if (!ans.contains(i))
                ans.add(i);
        return ans;
    }
}
```

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

Answer:

The first two are linear; that is, the running time is proportional to $n$ (a straight line with a positive slope). The intersection method is quadratic; that is, the plot of $n$ vs. $t$ is a parabola.

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.

Answer:

```java
public class Balanced {
  public static boolean isBalanced(String code) {
    String leftParens = "([{";
    String rightParens = "]})";
    Stack300<Character> parens = new Stack300<Character>();
    for (int i=0; i<code.length(); i++) {
      char next = code.charAt(i);
      int left = leftParens.indexOf(next);
      if (left != -1)
        parens.push(next);
      else {
        int right = rightParens.indexOf(next);
        if (right != -1) {
          if (parens.isEmpty())
            return false;
          else {
            char previous = parens.pop();
            if (leftParens.indexOf(previous) != rightParens.indexOf(next))
              return false;
          }
        }
      }
    }
    return parens.isEmpty();
  }

  public static void main(String[] args) {
    String ex1 = "void f() {int i=0; while (i<2) {System.out.println(i); }}";
    String ex2 = "void f() { int i=0; while (i<2 {System.out.println(i);}}";
    String ex3 = "void f() {int i=0; while (i<2) {System.out.println(i); }";
    System.out.println(isBalanced(ex1) + ": should be true");
    System.out.println(isBalanced(ex2) + ":应该为false");
    System.out.println(isBalanced(ex3) + ":应该为false");
  }
}
```