CMPS 11
Intermediate Programming
Midterm 2 Review Problems

1. Determine the output of the following Java program. Notice that the method `fcn2()` is overloaded, so there are really three distinct functions here.

```
// Problem1.java
class Problem1{
   public static void main( String[] args ){
      int a=5, b=3, c;
      double x=2.2, y=1.3, z;

      c = fcn1(a, b);
      b = fcn2(x, y);
      z = fcn2(c, b);
      System.out.println("a="+a+", b="+b+", c="+c);
      System.out.println("x="+x+", y="+y+", z="+z);
      System.out.println("bye!");
   }

   static int fcn1(int i, int j){
      int k = (i++) + (++j);
      return k+2;
   }

   static int fcn2(double u, double v){
      return fcn1((int)(u+v), 3);
   }

   static double fcn2(int r, int s){
      return (double)fcn1(r+s, 3);
   }
}
```

2. Fill in the definition of function `concatenate()` below. This function takes as input two int arrays *A* and *B*, of any length, and returns a new int array which is the concatenation of *A* and *B*, i.e. the length of the returned array is the sum of the lengths of *A* and *B*, and the contents of the returned array are the values in *A* followed by the values in *B*.

```
static int[] concatenate(int[] A, int[] B){
   // your code starts here









}  // your code ends here
```

3. Determine the output of the following Java program. Note that the function `f()` is recursive.

```
// Problem3.java
class Problem3{
   public static void main(String[] args) {
      System.out.println(f(7));
   }
   static int f(int n){
      System.out.println( "f(" + n + ")" );
      if(n<=1)
         return 2;
      else
         return f(n-1) + 5;
   }
}
```

4. Write a *recursive* function called `sum()` that takes a single `int` *n* as input and returns the sum of the positive integers from 1 to *n*. (Hint: this is the same as the recursive function that computes *n*! (*n* factorial) discussed in class, except with multiplication replaced by addition.)

```
static int sum(int n){. . . . }
```

5. Determine the output of the following Java program. (Hint: first figure out what each of the methods `f()`, `g()` and `h()` do, then put the pieces together.)

```
// Problem5.java
class Problem5{
   public static void main(String[] args){
      int[][] A = {{2, 4},{5, -1}};
      int[][] B = {{5, 3},{-3, 2}};
      System.out.println(f(A));
      h(B);
      System.out.println(f(g(B)));
      h(g(A));
   }
   static int f(int[][] M){
      return M[0][0]*M[1][1]-M[0][1]*M[1][0];
   }
   static int[][] g(int[][] M){
      int[][] N = new int[2][2];
      for(int i=0; i<2; i++)
         for(int j=0; j<2; j++)
            N[i][j] = M[j][i];
      return N;
   }
   static void h(int[][] M){
      for(int i=0; i<2; i++){
         for(int j=0; j<2; j++)
            System.out.print(M[i][j]+" ");
         System.out.println();
      }
   }
}
```

6. Determine the output of the following Java program. This program consists of two files Blah.java and BlahTest.java defining the classes Blah and BlahTest respectively.

```
// Blah.java
class Blah{
   private int foo;
   private double bar;
   Blah(int f, double b){
      foo = f;
      bar = b;
   }
   Blah(int a, int b){
      foo = a+b;
      bar = 0.0;
   }
   int getFoo(){
      return foo;
   }
   void setBar(double x){
      bar = x;
   }
   void mult(){
      bar *= foo;
   }
   public String toString(){
      return "("+foo+", "+bar+")";
   }
   public boolean equals(Object x){
      boolean eq = false;
      Blah B;
      if( x instanceof Blah ){
         B = (Blah)x;
         eq = (foo==B.foo && bar==B.bar);
      }
      return eq;
   }
}

// BlahTest.java
class BlahTest{
   public static void main(String[] args){
      Blah A = new Blah(15, 3.0);
      Blah B = new Blah(7, 8);

      System.out.println(A);
      System.out.println(B);
      A.mult();
      B.setBar(45.0);
      System.out.println(A.equals(B));
   }
}
```

7. Complete the methods `getMaxIndex()` and `getMinIndex()` in the Java program below. These function should return the index at which the maximum (respectively minimum) value of its array argument is stored. Functions `getMaxIndex()` and `getMinIndex()` should work on any `int` array of any length, not just the one specified in function `main()` below.

```
// Problem7.java
class Problem7{
   public static void main(String[] args){
      int [] list = {3, 9, 6, 12, 23, -25, 54, 9, 0, -12, 27};
      System.out.println(list[getMaxIndex(list)]);  // prints 54
      System.out.println(list[getMinIndex(list)]);  // prints -25
   }

   static int getMaxIndex(int[] A){
      // your code starts here









      // your code ends here
   }

   static int getMinIndex(int[] A){
      // your code starts here









      // your code ends here
   }
}
```

8. Determine the output of the following Java program. This program consists of two files: `Point.java` which defines the `Point` class, and `PointTest.java` which defines the `PointTest` class and contains function `main()`.

```
// Point.java
class Point{
   // Fields
   private int xcoord;
   private int ycoord;

   // Constructor
   Point(int x, int y){ xcoord = x; ycoord = y;}

   // Methods
   public String toString(){ return "(" + xcoord + ", " + ycoord + ")"; }
   void reflect(){ int temp = xcoord; xcoord = ycoord; ycoord = temp;}
   boolean isLeftOf(Point P){ return (this.xcoord < P.xcoord);}
   boolean isBelow(Point P){ return (this.ycoord < P.ycoord);}
}

// PointTest.java
class PointTest{
   public static void main( String[] args ){
      Point A = new Point(1, 5);
      Point B = new Point(2, -3);
      Point C = new Point(4, 3);
      Point D = new Point(8, 7);
      String str1, str2;

      A.reflect();
      D.reflect();
      System.out.println("A = " + A);
      System.out.println("B = " + B);
      System.out.println("C = " + C);
      System.out.println("D = " + D);
      str1 = A.isLeftOf(B)?"left":"right";
      str2 = C.isBelow(D)?"below":"above";
      System.out.println("A is to the " + str1 + " of B");
      System.out.println("C is " + str2 + " D");
   }
}
```

9. Write a Java class called `Distance` that encapsulates a single data field of type `double` representing a distance measured in Meters. Write a single constructor for the class that takes a `double` argument and copies its value to the data field. Write three access functions called `getMeters()`, `getFeet()` and `getFurlongs()` that return respectively: the value stored in the data field, the distance measured in feet and the distance measured in furlongs. (Google "feet per meter" and "furlongs per meter" to learn the relevant conversion factors.)

10. Write a function with the heading `static void printStringArray(String[] X)` that prints out the elements of its `String[]` argument `X` to `stdout` in reverse order, each on a separate line. In other words if `X` is the array `{one, two, three}`, then the output will be:

```
three
two
one
```

11. Write a function with the heading `static void sortStringArray(String[] X)` that sorts the elements of its `String[]` argument `X` in (increasing) alphabetical order. Thus if `X` is the array `{one, two, three, four}`, then a call to `sortStringArray()` on `X` will change it to `{four, one, three, two}`. Hint: follow one of the examples `BubbleSort.java` or `SelectionSort.java` discussed in class. To compare two `String`s as to their alphabetic order, use function `compareTo()` belonging to the `String` class (in the library `java.lang`.)