Final CS100J 12:00-2:30 Barton Hall East 13 May 2005

Question 0. _____	(out of 01)
Question 1. _____	(out of 12)
Question 2. _____	(out of 12)
Question 3. _____	(out of 12)
Question 4. _____	(out of 14)
Question 5. _____	(out of 12)
Question 6. _____	(out of 10)
Question 7. _____	(out of 15)
Question 8. _____	(out of 12)
Total _____	(out of 100)

The results of this final will be posted on CMS as soon as it is graded, hopefully this evening. Grades for the course may be available Saturday, but it may be Monday.

Please submit all regrade requests, using CMS where possible, BEFORE 6PM TONIGHT.

You have 2.5 hours to complete the questions in this exam, which are numbered 0..8. Please glance through the whole exam before starting. The exam is worth 100 points.

Question 0 (1 point). Print your name and net id at the top of each page. Please make them legible.

Question 1 (12 points). Loop. Write a loop (and its initialization) that stores the second largest number in **int** array `b` in variable `big2`. The precondition, invariant, and postcondition are given. You must use the given invariant.

```
// precondition: values in b are all different and b.length ≥ 2.




// invariant: big1 is the largest int in b[0..k],
//            big2 is the second largest int in b[0..k], and
//            0 < k < b.length
while (                                    ) {









}
//postcondition: big1 is the largest int in b[0..]  and
//               big2 is the second largest int in b[0..]
```

Question 2 (12 points). Arrays and methods. The first 5 rows of Pascal's Triangle are shown to the right. The first row is row 0, the second is row 1, etc.

```
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
……
```

The first and last values in each row are 1. In any row with more than two values, a value that is not 1 is obtained by adding the two entries diagonally above it. For example, in the Pascal Triangle on the right, 2 is the result of adding the two 1s diagonally above it. On the fifth row, each 4 is the result of adding the 1 and 3 (or 3 and 1) diagonally above it.

Write a method genPascalTriangle that generates a rectangular array that contains the first n rows of Pascal's triangle. Its specification is given below.

```
/** = a rectangular array b (say) in which each row k, 0 ≤ k < n,
      contains in b[k][0..k] the values in row k of Pascal's
      triangle.    Precondition: n ≥ 0 */
public static int[][] genPascalTriangle(int n) {
```

}

Question 3 (12 points). Matlab. (a) Assume `A` is a vector. Write a Matlab expression that counts the number of positive entries in A. **Do not write a loop.** Hint: you may use function `FIND`. Here is what you need to know about `FIND`.

```
    FIND(EXPR) evaluates logical expression EXPR and returns the
indices of those elements of the resulting matrix that are TRUE
(not 0). For example, FIND(A > 100) returns the indices of the
elements of A that are greater than 100. Thus, FIND([4 101 6 102
100] > 100) yields the array [2 4].
```

(b) The following series approximates the value of π. Write a Matlab function that, given a parameter `n`, `n > 0`, produces the sum of the first `n` terms of this series.

$$4 - 4/3 + 4/5 - 4/7 + \ldots$$

Question 4 (14 points). Executing Java statements. On this page are a class C and a class File.

(a) Fill in the 3 blanks in class File to truthify the definitions of the fields.

(b) Execute the call

C.doIt();

Write the output of println statements directly beneath the println statements, in the space provided.

```
public class C {
   public static void doIt() {
      File d= new File("movie1", 200);
      File f= d;
      d.expand(300);
      File e= new File("music", 100);
      d= e;
      e.expand(100);
      System.out.println("d: " + d.getSize() +
                         "\n e: " + e.getSize() +
                         "\n f: " + f.getSize() +
                         "\n avg: " + File.getAvgSize());









      File c1= new File("coursework", 100);
      File c2= new File("coursework", 50);
      c2.expand(50);
      System.out.println("c1 = c2: " + (c1 == c2));



      File g= e;
      e.expand(100);
      System.out.println("g: " + g.getSize() +
                        "\n avg: " + File.getAvgSize());



   }
}
```

```
public class File {
   private String name; // name of file
   private int size;       // size of file
   /* number of instances of File created */
   private static int numFile= 0;
   /* average size of all instances of File */
   private static double avgSize= 0.0;

   /** constructor: new File with name t, size m*/
   public File(String t, int m) {
      name= t;
      size= m;
      numFile= _____;

      avgSize= _____;
   }

   /** = size of this File */
   public int getSize() {
      return size;
   }

   /** = average size of files in the drawer */
   static public double getAvgSize() {
      return avgSize;
   }

   /** increase size of this File by m */
   public void expand(int m) {
      size= size + m;

      avgSize= _____;
   }
}
```

Question 5 (12 points). Strings. Assume Cornell Campus Mail decides to provide tracking service for interdepartment mail, for a fee. The tracking number is a 16 digit number stored in a String and encoded as follows:

Digits	**Purpose**
first	Service: 1 = express; 2 = regular
2nd- 6th	Sender account number
7th- 13th	Unique piece number
14th- 16th	Weight (in ounces, between 001-999)

Fee schedule:

express service postage = $3 + $.2 * weight(ounces)
regular service mail up to 16 ounces, $2 per piece
mail greater than 16 ounces, $2 + $.1 per extra ounce beyond 16 ounces

Write a method `cost` that produces the cost of the mail piece, given as parameter `t`.

The following methods in class Integer might be useful:

int	intValue()	= the value of this Integer as **int**
static int	parseInt(String s)	= s, converted to an **int**

```
/** = cost of mail piece t.
      Precondition: t is a valid tracking number */
static public double cost(String t) {




}
```

Question 6 (10 points). Abstract classes. The following abstract class represents storage devices used on a computer. Create a subclass `HardDrive` (you can put it on the next page). Note that you do *not* have to worry about the size of the content getting larger than the capacity of the hard disk. In addition to any components required from class `Storage`, `HardDrive` should contain the following fields and methods:

- **`double`** `used` (a field): The amount of space used on the drive.
- `Vector contents` (a field): contains the objects that are on the drive.
- **`void`** `addItem(Object obj, double s)`: A method for adding object `obj` of size `s` to the drive.
- **`void`** `format()`: A method to erase the contents of the HardDrive.

```
// Each subclass represents a different kind of Storage, all of which have a capacity
public abstract class Storage {

   private double capacity;  // capacity of this storage device

    /** Constructor: a new Storage with capacity c */
    public Storage(double c) {
        capacity=  c;
    }

    /** = the capacity of this Storage device */
    public double getCapacity() {
      return capacity;
    }

    /** = description of this Storage device */
    public String toString() {
      return "Storage device: " + capacity;
    }

    /** = amount of unused space */
    public abstract double remainingSpace();

}
```

This page is left intentionally nonblank. Use it for Question 6.

Question 7 (15 points). Subclass.

Part (a). Write a class `MP3Player` that extends subclass `HardDrive` of question 6. An `MP3Player` should contain an additional vector of `Strings` for the list of songs it contains and an integer for the song it is currently playing. If there is only one song in the list, that song is playing. Note that a song is part of the content of the hard drive and thus must appear in contents. Provide the following three methods (also add any fields that you think are necessary):

- A constructor `MP3Player(double c)`, where `c` is the capacity.
- `void addSong(String n, Object obj, double size)`: Add a song named `n` to the `MP3Player`. The actual data for the song is `obj`, which is of size `size`.
- `String getNextSong()`: start playing the next song (the one after the one that is currently playing) and return its name. If there are no songs, return **`null`**. If at the end the list of songs, return the first song.

Also, override method `format()` so that it removes information about songs as well as the content. You can write your class `MP3Player` on the next page.

Part (b). Apparent and real type. Consider the following declaration-assignments:

```
HardDrive x= new HardDrive(100.0);
MP3Player y= new MP3Player(200.0);
```

Write down what is printed by each of the following statement sequences. Assume that parts **a**, **b**, and **c** are **independent**, e.g. part **b** is executed immediately after executing the above declaration-assignments, and **not** after executing part **a.** If an error would result, say "BAD" and briefly explain the error.

a.
```
y.addSong("Lost At Sea", ..., 30.0);
x= y;
System.out.println(  x.getNextSong()  );
```

b.
```
y.addSong("Lost At Sea", ..., 30.0);
x= y;
System.out.println(  ((MP3Player)x).getNextSong()  );
```

c.
```
y.addSong("Lost At Sea", ..., 30.0);
Storage z= new Storage(30.0);
System.out.println(   z.toString()  );
```

This page is left intentionally nonblank. Use it for Question 7.

Question 8 (12 points). Algorithms.
Specify algorithm `Partition` by giving its pre- and post-conditions. Then, develop the algorithm. Do not write a complete method, with header and all; just write a sequence of statements to accomplish the task.