CS100J Spring 2005. Answers to final

1.
```
int k= 1;
int big1= Math.max(b[0], b[1]);
int big2= Math.min(b[0], b[1]);

// inv:big1 is largest in b[0..k],
//     big2 is 2nd largest in b[0..k],
//     0 < k < b.length
while (k != b.length - 1) {
  if (b[k+1] > big1) {
    big2= big1; big1= b[k+1];
  }
  else if ( b[k+1] > big2) {
    big2= a[k+1];
  }

  k= k + 1;
}
//post: big1 is largest in b[0..] and
        big2 is 2nd largest in a[0..]
```

2.
```
/** spec as on final */
public static int[][]
         genPascalTriangle(int n) {
   int[][] b= new int[n][n];
   // inv: rows b[0..k-1] have been
   //       calculated
   for (int k=0; k != n; k= k+1) {
     // calculate row k of triangle
       b[k][0]= 1;
       b[k][k]= 1;

       // inv: b[k][0..j-1] and
       //      b[k][k] have been
       //      calculated
       for (int j= 1; j < k; j= j+1) {
          b[k][j]=
            b[k-1][j-1] + b[k-1][j];
       }
   }

   return b;
}
```

3 (a)
```
length(find(A > 0))
```
(b)
```
function result = approxpi(n);
   nums= - ones(1, n);
   nums=   (-4) * cumprod(num);
   dens= [1:2:(2*n-1)];
   result= sum(nums ./ dens);
```

4 (a).
```
numFile= numFile + 1;
avgSize=
  (avgSize*(numFile-1) + m)/numFile;
avgSize=
  avgSize + ((double)m) / numFile;
```

(b)
```
d: 200
e: 200
f: 500
avg: 350.0

c1= c2: false

g: 300
avg: 250.0
```

5.
```
/** = cost of mail piece t.
       Precondition: t is a valid */
public static double cost(String t) {
  char service= t.charAt(0);
  int weight=
    Integer.parseInt(t.substring(13));
  if (service == '1') // express
     return 3 + 0.2 * weight;
  if (weight <= 16) return 2.0;
  return 2 + 0.1 * (weight - 16);
}
```

6.
```
public class HardDrive
                     extends Storage{

   private double used;//no bytes used
   private Vector contents;

   /** constructor: HardDrive with
                    capacity c */
   public HardDrive(double c) {
      super(c);
      format();
   }

   /** add item obj of size s */
   public void addItem(Object obj,
                       double s) {
      contents.add(obj);
      used= used + s;
   }

   /** erase this HardDrive */
   public void format() {
      contents= new Vector();
      used= 0;
   }

   /** = amount of unused space */
   public double remainingSpace() {
      return getCapacity() - used;
   }
}
```

7. part a.

```
public class MP3Player
                extends HardDrive {
  // songs in the player
  private Vector songs= new Vector();

  // song currently playing. -1 if
  // there are no songs
  private int current;

  /** constructor: a new MP3Player
      with capacity c */
  public MP3Player(double c) {
    super(c);
  }

  /** erase content and song list */
  public void format() {
    super.format();
    songs= new Vector();
    current= -1;
  }

  /** add song obj with name n and
      size s */
  public void addSong(String n,
              Object obj, double s) {
    addItem(obj, s);
    songs.add(n);
    if (current == -1) current= 0;
  }

  /** play the next song and return
      its name (null if no songs) */
  public String getNextSong() {
    if (current==-1) return null;
    current=
      (current + 1) % songs.size();
    return (String)songs.get(current);
  }
 }
```

7 Part b.

a. BAD. Apparent class of x is HardDrive, and getNextSong is not accessible in HardDrive.

b. "Lost At Sea"

c. BAD. Cannot create instance of an abstract class.

8. Algorithm Partition. See class notes.