CS125 Second Examination Spring 2012 SOLUTION

1. Short Answers – 15 points (3 points each)

```
Student s1, s2, s3;
s1 = new Student ("A", 5,4);
s2 = new Student ("B", 5,4); s1,s3
s3 = new Student ("C", 2,3); s2
s1 = s2;
s2 = s3;
s3 = s1;
System.out.print(s3 == s2);
```

false

```
System.out.print(s1 == s2);
```

false

(b) Briefly explain why no Tweet objects are created in the following code.

```
Tweet[] t = new Tweet[10];
TextIO.putln(t == null);
```

Write your explanation here: **Creates an array of pointers to Tweets**

(c) Is Java's String method *indexOf* =**instance**, Math.max = **class** method.

(d) Your friend incorrectly claims Point objects are immutable (see class definition below). Write the line numbers of all of the lines inside the Point class definition that cause Pt objects to be mutable.

```
 public class Point {
2  public int x,y;
3  public String asLabel() {return "("+x+","+y+"):"+count;}
4  public int getX() {return x;}
5  public int getY() {return y;}
6  public void increment() { x++; y--; }
7  public boolean isValid() {return y>0; }
8  public void setValues(int[] arr) {x=arr[0];y=arr[1];}
9  public Point(int xx, int yy) { x=xx; y=yy; count++;}
10  public Point(Point src) { x=src.x; y=src.y; count++;}
11  public static int count=0;
 }
```

Mutability due to (Line number(s)) : **2,6,8**

(e) Using the Point class above, what will the following program print when executed? Also determine the new output if the 'static' modifier is deleted from '`int count`' (line 11) and added to '`int x,y`' (line 2)

	Original output	Output after lines 2 & 11 changed

`Point a = new Point(10,20);`
`Point b = new Point(30,40);`
`b.increment();`
`System.out.println(a.asLabel());`
`System.out.println(b.asLabel());`
	(10,20):2
	(31,39):2
		(31,39):1
		(31,39):1

2. My Sketch items. – 16 points

```
public class SketchList {
  private Point[] list  = new Point[0];

  public int getSize()                        {return list.length; }

  public Point getPoint(int index)     {return list[index]; }

  public void add(Point p) {
       Point[] tmp = new Point [list.length + 1];
    for(int i =0;i<list.length;i++)
         tmp [i] = list[i];
       tmp [tmp.length -1] = p;
       this.list = tmp;
  }
```

/* Returns a new SketchList object with points from this original list at the requested (x,y) position. Hint: Use Point's *getX*,*getY* methods and the above *add* method to construct the result. */

```
  public SketchList find(int x, int y) {

      SketchList result = new SketchList();
        for(int i = 0; i < list.length; i++) {
          Point p = list[i];
          if(p.getX() == x && p.getY() == y) result.add(p);
        }
        return result;
      }
    }
```

Returns the first (zero[th] index) item in the list and also removes it from the list i.e. the list size is reduced by one. If the list is already empty this method returns null and does not modify the list.

```
public Point removeFirst() {
    if(list.length == 0) return null;
    Point[] tmp = new Point [list.length - 1];
    for(int i = 1; i < list.length; i++)
       tmp [i-1] = list[i];
    Point p = list[0];
    this.list = tmp;
    return p;
}
```

Returns *true* iff any points in this list are at position (0,0), *false* otherwise. Use Sketchlist's *find* and *getSize* methods, specified above, to implement this method with minimal code (one or two lines).

```
public boolean anyPointsAtOrigin() {
    return find(0,0).getSize() == 0;
  }
} // end class
```

3. I took CS125 and all I got was this startup company and nerdy friends – 20 Points

(a) Write code below so that `extract` returns an sub-image, represented as a 2D array. Return the pixels in the rectangle that starts at (*x1*,*y1*) and is of size (*w* by *h*) pixels.

- If the *src* parameter is null return a 2D integer array of size 1x1.
- Pixels in the output image that do not map to a valid source pixel should be grey (rgb value 0x808080).
- You may assume *x1*,*y1*,*h*,*w* are non-negative (i.e. >=0).

```
public static int[][] extract(int[][] src, int x1, int y1, int w, int h) {
  if(src == null) return new int[1][1];
  int[][] result = new int[w][h];
  for(int x=0; x<w; x++)
      for(int y=0; y<h; y++) {
          int srcX = x+x1;
          int srcY = y+y1;
          boolean inside = srcX < srcWidth && srcY < srcHeight;
          // oneliner... result[x][y] = inside ? src[srcX][srcY] : 0x808080;
          if(inside) result[x][y] = src[srcX][srcY];
          else result[x][y] = 0x808080;
       }
  return result;
};
```

(b) Complete the description of the following image transformation. Circle the correct responses.

```
if((src[x][y] & 0xff)>200) output[x][y]=0xff0000; else output[x][y]=0x00ff00;
```

"For each pixel, when the { **BLUE** } component is greater than 200,
the output is { **RED** } otherwise it is { **GREEN** }"

(c) Write an expression that returns true if the blue component is greater than the red component for 'rgb' – the color encoded as an integer of a single pixel.

```
public static boolean moreBlueThanRed(int rgb) {
    return (rgb & 0xff)  > (rgb >>16);
}
```

4. White-collar crime: Testing the de-shredder – 18 Points

```
public class Document {
    private Strip[] pieces;
    public Document(Strip[] p) { this.pieces = p;}
    public Document(Document src) {
        pieces = new Strip[ src.pieces.length ];
        for(int i =0; i < pieces.length; i++) {
            pieces[i] = new Strip( src.pieces[i] );
        }
    }
    public void shuffle(int N) {
        for(int i =0; i < N;i++) {
            int x = (int)(Math.random() * pieces.length);
            int y = (int)(Math.random() * pieces.length);
            Strip s = pieces[x]; pieces[x] = pieces[y]; pieces[y] = s;
        }
    }
    public int score() {
        int count =0;
        for(int i=1; i <pieces.length; i++)
            if(pieces[i].getId() == pieces[i-1].getId() + 1) count ++;
        return count;
    }
```

5. My Deshredding Simple Genetic Algorithm is too good for CSI – 18 Points

The evidence is shredded into 25 vertical strips. Create the following class methods in Deshredder.java. Do not create any other methods or instance variables or class variables. You may call other specified methods in this class, the *Strip* and *Document* classes (see Q4), and you may create temporary variables.

1. Create a public class method named '*create*' that takes no parameters and returns a new *Document*:
   a. First create an array of 25 *Strip* objects for use by the new *Document*. Create each shred with a unique identifier (0... 24), then initialize a new document with this array.
   b. Before returning the document, use the Document's *shuffle* method to ensure all of the document's strips are in a random sequence.
2. Create a public class method named '*mutate*' that takes an array of *Document* objects (as specified in Q4) and two integers 'i' and 'j'. Do not return anything. Assume 'i' and 'j' refer to valid non-null array entries.
   a. Replace the j^{th} entry with a new, *deep* copy of the i^{th} document.
   b. Also use call *shuffle (see Q4)* on this new j^{th} entry to perform one random swap.
3. Create a public class method called '*main*' that takes a string array and does not return anything.
   a. Use the *create* method above to create 10000 Documents and store them in an array.
   b. Print out the *array index* of a document with the highest score of all documents. You only need to print one index, even if multiple documents have the same highest score.

```
public class Deshredder {
  public static Document create() {
    Strip[] arr = new Strip[25];
    for(int i = 0; i < arr.length; i++)
        arr[i] = new Strip(i);
    Document d = new Document(arr);
    d.shuffle(25);
    return d;
    }
  public static void mutate(Document[] docs, int i, int j) {
    docs[j] = new Document( docs[i] );
    docs[j].shuffle(1);
  }
  public static void main(String[] args) {
    Document[] docs = new Document[10000];
    for(int i = 0; i < docs.length; i++)
        docs[i] = create();
    int best = 0;
    for(int i = 0; i < docs.length; i++)
        if(doc[i].score() > doc[best].score())
            best = i;
    System.out.println(best);
  }
}
```

6. Concepts and code snippets – 13 Points (Last Question)

(a) Write one line after the following code: Use variable '*ss*' to cause a *NullPointerException* to be thrown.

```
String[] ss = new String[10];
ss[0].toUpperCase();
```

(b) Identify the following use of three data structures a,b and c, as examples of a stack, map, queue, or tree:

```
a.push (3);
a.push(4);
a.pop(); // returns 4
a.pop(); // returns 3
```
STACK

```
b.add(3);
b.add(4);
b.next(); // returns 3
b.next(); // returns 4
```
QUEUE

```
c.add(100,"OK");
c.add(404,"NOT FOUND");
c.find(100); //returns "OK"
c.exists(299); //returns false
```
MAP

(c) Briefly describe the purpose of single-step debugging ("SSD"): Your answer should include what a programmer might verify while stepping through code.

Verify program's variable values (i.e. it's state) ; verify program's code path (e.g. which branches are taken, how many iterations) as the virtual machine steps through each line of code.

(d) An iterator is a class that provides simple sequential access to elements of a data structure. An example iterator is shown below.

```
class ArrayIterator {
        private double[] values;
        private int position=0;

        public ArrayIterator (double [] data) {values = data;}
        public boolean hasNext() {return position<values.length;}
        public double next() { return values[ position ++ ];}
}

public static void main(String[] args) {
    double[] data = {10.5, 17.5, 31.3, 42.3};

    ArrayIterator it = new ArrayIterator(data)
    while( it.hasNext() ) {
            System.out.println( it.next() );
}
```