Name _____

CS 201
Spring 2016
Instructor: Heather Pon-Barry

Practice Midterm Exam

General description

This is a **closed book, closed notes, and closed computer** exam.

There are 5 problems, some subdivided into smaller questions. Do as many as you can, or as much of a bigger problem as you can. There should be enough time to finish, or at least to start working on, all the questions. Feel free to use extra paper or the backside of the paper, but be sure to put your name and which question is being answered at the top. Start with the ones that you consider to be easiest, then move to those that may take more time.

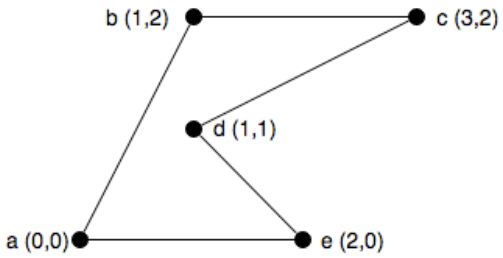
All code is and should be in **Java**! Be generous with comments. Part of the grade will be based on how readable and neat your code is. You may use **`S.o.p`** as an abbreviation for **`System.out.println`**.

Question	Points
1	/ 25
2	/ 20
3	/ 20
4	/ 15
5	/ 20
Total	**/ 100**

Name _____

Please use extra paper as needed to write your answers for Question 1!

1. ***Debugging*** (25 points)
   A 2D *polygon* is defined by a sequence of *n* points in the plane (each point has an *x* and a *y* coordinate). For instance, the polygon below is defined by the sequence of points (a, b, c, d, e); the sides are defined by "connecting the dots" in order, returning to the first to close the polygon.



   On the following page you have been provided with a class `Polygon` that represents a 2D polygon; the **constructor** should expect the coordinates of the points to be passed to the constructor as a 2D int array.

   The class should define the following public methods:
   - **getNumPts()** - returns the number of points defining the polygon
   - **printNumPts()** - prints the number of points defining the polygon (e.g., for the example above, this would print *"The Polygon has 5 points."*)
   - **computePerimeter()** - returns the *perimeter* of the polygon

   Useful information:
   The perimeter is calculated by adding the lengths of all the sides of the polygon.
   The distance between two points (x1, y1) and (x2, y2) is

   $$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

   The Math class provides a function for computing the square root:

```
Math.sqrt( double d );
```

   Your Task:
   You have been tasked with fixing problems with the Polygon class. These problems have reported to you as follows:
   - the number of points printed is incorrect
   - the perimeter calculation is close but incorrect

Name _____

Polygon Source Code

// in Polygon.java

```
public class Polygon
{
    /* holds the points of the polygon
     * in the format point n, x = 0, y = 1
     * x = polyArray[n][0]
     * y = polyArray[n][1]
     */
    private int[][] polyArray;

    public Polygon( int[][] array ) {
        polyArray = array;
    }

    /* getNumtPts()
     * returns the number of points
     * defining the polygon.
     */
    public int getNumPts() {
        return polyArray.length;
    }

    /* getX - returns the x
     * offset of the nth point
     * defining the polygon
     */
    public int getX(int n) {
        return polyArray[n][0];
    }
    /* getX - returns the y
     * offset of the nth point
     * defining the polygon
     */
    public int getY(int n) {
        return polyArray[n][1];
    }

    /* printNumPts()
     * prints the number of points
     * "The Polygon has 5 points."
     */
    public void printNumPts() {
        System.out.println("The Polygon has " + getNumPts()-1
                                               + " points.");
    }
```

Name _____

```
  /* computePerimeter()
   * returns the perimeter of the polygon
   * stored in the interal array
   */
    public double computePerimeter() {
        double perimeterSum = 0.0;

        double deltaX;
        double deltaY;
        double distTemp;

        for( int n = 1; n < getNumPts(); n++ ) {
            deltaX = getX(n) - getX(n-1);
            deltaY = getY(n) - getY(n-1);
            distTemp = (deltaX * deltaX) + (deltaY * deltaY);
            double dist = Math.sqrt( distTemp );
            perimeterSum += dist;
        }
        return perimeterSum;
    }

    public static void main(String[] args) {
        int[][] array = new int[4][2];
        array[0][0] = 0; array[0][1] = 0;
        array[1][0] = 0; array[1][1] = 2;
        array[2][0] = 2; array[2][1] = 2;
        array[3][0] = 2; array[3][1] = 0;

        Polygon myPolygon = new Polygon( array );
        myPolygon.printNumPts();
        System.out.println("This poly has " +
             myPolygon.computePerimeter() + " perimeter");
    }
}
```

Faulty Output:

```
The Polygon has 4 points.
This poly has 6.0 perimeter
```

Name _____

2. ***Sorting*** (20 points)

Given the following array of numbers, show the first five steps of how the following sorting algorithms will sort the array. Consider one step to be when there is a change to the order of the elements in the array (each step should have all seven elements in the array).

a) Bubble sort

`Initial array`	18	9	11	24	16	2	8

`Step 1`							
`Step 2`							
`Step 3`							
`Step 4`							
`Step 5`							

b) Selection sort

`Initial array`	18	9	11	24	16	2	8

`Step 1`							
`Step 2`							
`Step 3`							
`Step 4`							
`Step 5`							

c) Insertion sort

`Initial array`	18	9	11	24	16	2	8

`Step 1`							
`Step 2`							
`Step 3`							
`Step 4`							

Name _____

Step 5 | | | | | | | |

3. ***Function execution and Recursion*** (25 points)
   a. ***Function execution*** (10 points)
   Given the following code, what will be printed out? Be sure to show how you arrived at your answer (i.e., show the function calls).

```
1  public class ExampleClass {
2      public static int foo( int a, int b ) {
3          if ( (a == 0) || (b == 0) )
4              return a + b;
5          else
6              return foo(a-1, b-1) + foo(a-1, b);
7      }
8
9      public static void main( String[] args ) {
10         System.out.println( foo(3,2) );
11     }
12 }
```

Name _____

b. ***Recursion*** (15 points)
Write a recursive function to compute the number of primes between integers *a* and *b*, inclusive, using the following algorithm:

$$count\_primes(a,b) = \begin{cases} 0 & \text{if } a > b \\ count\_primes(a, b-1) + 1 & \text{if } isPrime(b) \\ count\_primes(a, b-1) & \text{otherwise} \end{cases}$$

You may assume that you have access to a function with the following call signature:

```
/**
 * Returns true if the number is prime and false
 * otherwise.
 */
public boolean isPrime( int num );
```

Name _____

4. ***Binary search (15 points)***

Recall the binary search algorithm:

```
public static int binarySearch( int[] sorted, int test )
{
	return binarySearch( sorted, test, 0, sorted.length-1 );
}

private static int binarySearch( int[] sorted, int test, int lo, int hi )
{
	if ( hi < lo )
		return -1;
	else
	{
		int mid = (hi + lo)/2;
		if ( test == sorted[mid] )
			return mid;
		else if ( test < sorted[mid] )
			return binarySearch( sorted, test, lo, mid-1 );
		else
			return binarySearch( sorted, test, mid+1, hi );
	}
}
```

continued on next page…

Name _____

Given the following array, `y`, show how the binary search algorithm performs when searching for the values below. Indicate the **index bounds** (lo, hi, mid) for each recursive invocation of the binary search method and the **value returned**.

y	1	3	15	22	30	32	41	53	61	65	67	77	82	84	91	95	102
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

a) `binarySearch( y, 53 )`

LO	HI	MID

return value:

b) `binarySearch( y, 65 )`

LO	HI	MID

return value:

c) `binarySearch( y, 81 )`

LO	HI	MID

return value:

d) `binarySearch( y, 0 )`

LO	HI	MID

return value:

Name _____

5. ***Objects and program execution (20 points)***
What gets printed out by the following code?
(You should sketch out memory with any modifications.)

```
BiggerMystery m;
m = new BiggerMystery();
System.out.println( "m has value " + m.toString() );
```

You are given the following Java class definitions.

Name _____

---------- **Mystery.java** ----------

```
public class Mystery
{
   protected int val;
   protected Thing t;
   public Mystery() {
         val = 1;
         t = new Thing( this );
   }
   public void accum( int x ) {
         val *= x;
   }
   public String toString() {
         return Integer.toString( val );
   }
}
```

---------- **BiggerMystery.java** ----------

```
public class BiggerMystery extends Mystery
{
   public BiggerMystery () {
         super();
   }
   public void accum( int x ) {
         super.accum( x );
         val += x;
   }
}
```

---------- **Thing.java** ----------

```
public class Thing
{
   private int[][] arr;
   private Mystery mystery;
   public Thing ( Mystery mystery ) {
         this.mystery = mystery;
         arr = new int[2][3];
         for ( int i = 0; i < arr.length; i++ )
               for ( int j = 0; j < arr[i].length; j++ )
                     arr[i][j] = (i+1)*j;
         doYerThing();
   }
   private void doYerThing() {
         for ( int i = 0; i < arr.length; i++ )
               for ( int j = 0; j < arr[i].length; j++ )
                     mystery.accum( arr[i][j] );
   }
}
```