You may use <u>one</u> 8.5"x11" page of <u>handwritten</u> notes (dual-sided ok), but no books, computers, calculators, or cell phones during the exam. (Use this to help prepare for the exam! Include details that you may not remember during the exam)

CIS 200 Practice Problems for Final

<u>This is NOT a past final exam</u>. This is simply several practice problems to review <u>*some*</u> of the <u>concepts</u> that may appear on the exam. These practice problems cover <u>only</u> the concepts since Exam 3. I would strongly suggest you also carefully review all practice and actual given exams (1-3)

1) Write the **Java** class Dictionary. This class should contain the following:
   - An array of strings (words) as an **instance variable**
   - A **constructor** that takes an array of strings as a parameter, creates the necessary space for the instance variable array, and copies the values from the parameter into the instance variable.
   - A **containsWord method** that takes a string as a parameter. This method should return true if the parameter is in the instance variable array, and false otherwise.
   - A **startsWith method** that takes a character as a parameter. This method should return a count of how many words in its instance variable array start with that parameter character.

```
public class Dictionary {
    private String[] words;

    public Dictionary(String[] arr) {
        words = new String[arr.length];
        System.arraycopy(arr,0,words,0,arr.length);
                    (can also use a for loop to copy element by element)
    }

    public boolean containsWord(String s) {
        for (int i = 0; i < words.length; i++) {
            if (words[i].equals(s)) return true;
        }

        return false;
    }

    public int startsWith(char c) {
        int count = 0;
        for (int i = 0; i < words.length; i++) {
            if (words[i].charAt(0) == c) count++;
        }

        return count;
    }
}
```

2) Create a Java class called **TestDictionary** that contains a **main** method. Inside the main method, declare a Dictionary object with the words “apple”, “banana”, and “avocado”, and “carrot”. Print how many words begin with the letter ‘a’. You must call the startsWith method when making your calculations.

```
public class TestDictionary {
    public static void main(String[] args) {

    String[] arr =
            {“apple”,”banana”,”avocado”,”carrot”};

    Dictionary d = new Dictionary(arr);

    int result = d.startsWith(‘a’);

    System.out.println(result);

    } // end main
} // end class
```

3) Write the following **Java** method:

public static int *indexOf*
(String[] words, String s)

This method should return the index of s within the words array. If s is not an element in words, you should **throw an appropriate exception**.

What is being passed into the method? Out of? How would you do it "manually" (i.e. w/o computer)

```
public static int indexOf (String[] words, String s) {
        for (int i = 0; i < words.length; i++) {
                if (words[i].equals(s))
                        return i;
        }

        throw new IllegalArgumentException();
}
```

4) Write the **C#** class Point, which represents an (x,y) coordinate. It should have (double) **instance variables** for the x and y values, and a **constructor** that takes parameters for the x and y values and initializes the instance variables.

It should have the method **Print**, which prints this point in the format **(x,y)**. Lastly, it should have the method **Midpoint**. The Midpoint method should take another Point as a parameter, and should return a new Point object that is the midpoint between THIS point and the parameter. For example, the midpoint between (1,3) and (5,2) is ([1+5]/2,[3+2]/2) = (3, 2.5)

```
public class Point {
        private double x;
        private double y;

        public Point(double a, double b) {
                x = a;
                y = b;
        }

        public void Print() {
                Console.WriteLine( "(" + x + "," + y +")" );
        }

        public Point Midpoint(Point another) {
                double newX = (x+another.x)/2;
                double newY = (y+another.y)/2;
                Point result = new Point(newX,newY);
                return result;
        }
}
```

5) Write a **C#** class called TestPoint. This class should contain a Main method. Inside Main, do the following:
   a) Declare an array of type Point that can hold 10 points.
   b) Use a loop to ask the user to enter (x,y) coordinates for each point. For each one, create a new Point object with that information and store it at the current array location.
   c) Print the first point in the array by calling its Print method.
   d) Get the midpoint between the second and last points by calling the Midpoint method.
   e) Print that midpoint by calling its Print method.

```
public class TestPoint {
  public static void Main() {
        Point[] points = new Point[10];

for (int i = 0; i < 10; i++) {
        Console.Write("Enter x: ");
        double x = Convert.ToDouble(Console.ReadLine());
        Console.Write("Enter y: ");
        double y = Convert.ToDouble(Console.ReadLine());

        points[i] = new Point(x,y);
        } // end for

        points[0].Print();

        Point mid = points[1].Midpoint(points[9]);
        mid.Print();
  } // end main
} // end class
```

6) Consider the C# program below. This program is supposed to get 10 numbers from the user, sort them (from smallest to largest), and then print them out in sorted order. However, the program has a mistake that prevents it from working properly. What is the mistake? What effect will this mistake have on the final list of numbers that is printed? How might the program be fixed to work correctly?

```
using System;
public class FinalExam {
        public static void Main() {
                int[] nums = new int[10];
                for (int i = 0; i < nums.Length; i++) {
                        Console.Write("Enter number: ");
                        nums[i] =
                                Convert.ToInt32(Console.ReadLine());
                }

                int pos;
                for (int i = 0; i < nums.Length; i++) {
                        pos = i;
                        for (int j = i+1; j < nums.Length; j++){
                                if (nums[j] < nums[pos]) pos = j;
                        }
                        Swap(nums[i], nums[pos]);
```

```
            }

            for (int i = 0; i < nums.Length; i++) {
                  Console.Write(nums[i] + " ");
            }
            Console.WriteLine();
      }

      public static void Swap(int x, int y) {
            int a = x;
            x = y;
            y = a;
      }
}
```

1) First, what does the first loop do?

(Reads in 10 int)

2) What does nested for loop attempt to do?

(Find index of the smallest number and move to position 'i' … sort and exchange sort)

3) Anything wrong with the logic?

Swap is past by value; nothing in array is actually swapped

4) What will be displayed?

It will print original array

5) Solution:

Could pass in array and both indices.

7) Consider the Point class from #4. In each of the following statements, explain what it does, what is printed, or if there is an error. Assume the statements are executed in order.

a) **Point p1 = null;**

Declares p1-currently has no value (null)

b) **Point p2 = new Point(3.0,4.2);**

Creates the point (3.0,4.2)and p2 references this same point

c) **double[] vals = {2.5,3.1};**
Point p3 = new Point(vals);

Won't compile – Point constructor needs two doubles, not an array.

d) **Point p4 = new Point(0.0,0.0);**

p4 references the new Point (0.0,0.0)

e) **p2.Print();**

Displays: (3.0,4.2)

f) **Console.WriteLine(p2);**

Displays memory address of *object* p2

g) **Point p5 = p2;**
p5.Print();

p5 now references the same point as p2 (3.0,4.2) … displays: (3.0,4.2)

h) **p5 = new Point(4.8,5.6);**

Creates a new Point object (4.8,5.6)
p5 now references that new point

i) **p2.Print();**

Displays: (3.0,4.2)

j) **p5.Print();**

Displays: (4.8,5.6)

k) **p2 = p2.Midpoint(p5);**
p2.Print();

p2 is now the point (3.9,4.9)
Displays: (3.9,4.9)

l) **Console.WriteLine(p4.Midpoint(p2));**

Prints the memory address of the midpoint
(Midpoint method returns object reference)

m) **int val = 5;**
p1 = p5.Midpoint(vals);

Won't compile – Midpoint needs a Point object as a parameter

n) **p1.Print();**

NullReferenceException – p1 is null