PROBLEM 1 : (***What is put out? (30 points)***)

A. (14 pts) List the output for the following code.

```
int x = 5;
double y = 2.0;
int z = 3;
String phrase = "Will Durham get 54 inches of snow?";

System.out.println("x/z = " + x/z);
System.out.println("x/y = " + x/y);

System.out.println("multiply " + x*z);
System.out.println(Math.floor(6.7));

System.out.println(phrase.substring(6,9));
System.out.println(phrase.substring(phrase.indexOf("snow")+1));

int [] values = {4, 9, 12, 3, 6, 7};

for (int k = 1; k < values.length; k++)
{
  if (values[k] > values[k-1] )
  {
    System.out.print(values[k] + " ");
  }
}
System.out.println("done");
```

List OUTPUT here:

B. (10 pts) Consider the following `Mystery` method.

```
public int Mystery(Scanner in, String status)
{
   int count = 0;
   while (in.hasNext())
   {
      String name = in.next();
      int age = in.nextInt();
      String condition = in.next();
      if ( age < 13 && condition.equals(status))
```

```
        {
            count++;
        }
    }
    return count;
}
```

Answer the following questions about this code.

1. What is the **return type** for the method Mystery?

2. What are the **names** of the parameters in the Mystery method?

3. What are the **types** of the parameters in the Mystery method?

4. Assume the following data file has been bound to the Scanner in and is ready for reading.

   ```
   Maia 11 critical
   Sarah 15 stable
   Jeffrey 3 stable
   Quiang 8 critical
   Yu 11 critical
   ```

   What is the return value of the call `Mystery(in, "critical")`, where `in` is a Scanner bound to the above data file?

5. Explain what the method Mystery does.

C. (6 pts) Consider the following graphics code. Remember that the top left corner of the drawing canvas is (0,0), the x values increase to the right and the y values increase as they go down. The method `fillRect(int x, int y, int width, int height)` draws a rectangle filled in with the current color and with its top left corner at (x,y).

```
Point myCenter = new Point(50, 100);
int width = 50;
int height = 100;

pen.setColor(java.awt.Color.red);
pen.fillRect(myCenter.x, myCenter.y, width, height);
```

```
myCenter.x += 50;
myCenter.y += 100;

pen.setColor(java.awt.Color.green);
pen.fillRect(myCenter.x, myCenter.y, width, height);
```

Draw a picture showing what this code draws. Identify the color and top left x and y coordinates of any shape in your drawing.

PROBLEM 2 : (*Numbers everywhere (12 points)*)

Write the method `averageInRange` that returns a double and has three parameters: an integer array called numbers, an integer value named `low` and an integer value named `high`. This method finds the average of the integers in `numbers` that are between low and high, inclusive.

For example, if the array `elements` contained the values: 9, 3, 5, 1, 15, 18, 11

Then the call `averageInRange(elements, 9, 16)` would return 11.66, the average of 9, 11 and 15. For the same array, the call `averageInRange(elements, 2, 14)` would return 7.0, the average of 9, 3, 5 and 11.

Complete the method below.

```
public double averageInRange(int [] numbers, int low, int high)
{
```

PROBLEM 3 : (*Birthday celebration (14 points)*)

Write the method `findNames` which has two parameters: an ArrayList of Strings named `birthdates` that have names and birth dates in the format "name: month day, year", and an integer named `year`. Note that `name` may be any number of words. This method returns an ArrayList of the names of those people born in `year`.

For example, suppose `birthdates` contained the following five strings:

```
"Jessica Chang: Feb 10, 1991"
"Leo Rofe: Aug 3, 1973"
"Chris Brown: May 14, 1991"
"Wayne Dark Light: Dec 25, 1985"
"Zhiyi Zhang: Nov 24, 1995"
```

Then the call findNames(birthdates, 1991) would return an Arraylist with the two entries of people born in 1991: `"Jessica Chang", "Chris Brown"`.

Complete the method below.

```
public ArrayList<String> findNames(ArrayList<String> birthdates, int year)
```

PROBLEM 4 : (*The Olympics go on (26 points)*)

Consider the SkiJumper class shown below to store information about Olympic Ski jumpers. Ski jumping is a sport in which skiers go down a ramp and then attemp to fly as far as possible before landing. The person who flies the furthest distance is the winner.

```
public class SkiJumper {

  private String myName;        // name of ski jumper
  private double myCurrentJump; // current jump distance
  private double myBestJump;    // best jump distance so far

  // constructor - input is the name of a skier and their first jump
  public SkiJumper(String name, double jump) {   // code not shown
  }

  // returns name of SkiJumper
  public String getName(){   // code not shown
  }

  // returns current jump value
  public double getCurrentJump()   {  // code not shown
  }

  // returns best jump value
  public double getBestJump()   {  // code not shown
  }

  // set the current jump distance to jump. Set best jump distance if
  //    this is the best jump so far
  public void setCurrentJump(double jump)   {  // code not shown
  }
}
```

PART A. (10 pts) Fill in the missing code in the SkiJumper methods below.

```
  // constructor - input is the name of a skier and their first jump
  public SkiJumper(String name, double jump)
  {
```

```
}

// returns name of SkiJumper
public String getName()
{



}

// returns current jump value
public double getCurrentJump()
{



}

// returns best jump value
public double getBestJump()
{


}

// set the current jump distance to jump. Set best jump distance if
//    this is the best jump so far
public void setCurrentJump(double jump)
{







}
```

PART B. (16 pts) Write the following method called `competition` that has one parameter of type Scanner named `in`. This method runs a ski jump competition between two skiers, then prints out the longest jump for each skier and a message as to who jumped the furthest.

In particular, the Scanner is ready to read from a file that has data in the following format on each line: `firstname lastname amount`, where `firstname` and `lastname` are each one word with no blanks, and `amount` is the distance the skier has jumped. The first two lines

are the results of the first jumps for the two skiers. The remaining lines are additional jumps of the skiers and may be in any order and it is unknown how many additional jumps they each will complete. There will be at least two lines in the file with different names. There are only two skiers. You can assume they jump enough times that one of them jumps further than the other (there will be a clear winner).

Here is a sample data file (your program should work with other data files).

```
Simon Ammann  98.0
Adam Malysz   94.5
Simon Ammann  104.5
Adam Malysz   105.0
Adam Malysz   103.5
Simon Ammann  108.0
Simon Ammann  102.5
Adam Malysz   104.0
```

Here is sample output for this data file:

```
Simon Ammann's longest jump was 108.0
Adam Malysz's longest jump was 105.0
Simon Ammann jumped further
```

Here is an outline of what you should do:

1. Create two skiers and process all the data in the datafile. (assume the Scanner `in` is bound to a file and ready for reading.)
2. Print the longest jump for each skier and a message indicating who jumped the furthest.

Complete this method on the next page.

```
public void competition(Scanner in)
{
```

Assume that the following classes and methods are available.

```
public class String {
    // Returns the length of this string.
    public int length ()
    // Returns a substring of this string that
    // begins at the specified beginIndex and
    // extends to the character at index
    // endIndex - 1.
    public String substring (int beginIndex,
                             int endIndex)
    // Returns a substring of this string that
    // begins at the specified beginIndex and
    // extends to the end of the string.
    public String substring (int beginIndex)
    // Returns position of the first
    // occurrence  of str, -1 if not found
    public int indexOf (String str)
    // Returns the position of the first
    // occurrence of str after index start
    // returns -1 if str is not found
    public int indexOf (String str, int start)
    // returns character at position index
    public char charAt(int index)
    // returns true if str has the exact
    // same characters in the same order
    public boolean equals(String str)
    // returns the string as an array
    // of characters
    public char [] toCharArray()
}

public class Arrays {
    // Sorts the specified array into
    // ascending numerical order
    public static void sort(int[] a)
}

public class Integer {
    // Returns the argument as a signed integer.
    public int parseInt(String s)
}

public class Double {
   // Returns the argument as a double
   public double parseDouble(String s)
}
```

```
public class Random {
    // Create a new random number generator
    public Random()
    // Returns a pseudorandom, uniformly
    // distributed value in [0,n)
    public int nextInt(int n)
}

public class ArrayList {
    // Constructs an empty list
    public ArrayList ()
    // Returns the number of elements
    public int size ()
    // Returns element at position index
    public Object get (int index)
    // Replaces the item at position index
    // with element.
    public Object set (int index, Object element)
    // Appends specified element to end of
    // this list.
    public boolean add (Object o)
}

public class Scanner
{
    // Create Scanner that reads data from a file.
    public Scanner (File file)
    // Create Scanner that reads data from a string.
    public Scanner (String str)
    // Change delimiters used to separate items
    public void useDelimiter (String characters)
    // Check if more items are available
    public boolean hasNext ()
    // Get next delimited item as a string
    public String next ()
    // Get next line as a string
    public String nextLine ()
    // Get next delimited item as an integer value
    public int nextInt ()
    // Get next delimited item as a Double value
    public double nextDouble ()
}
```