1. **Short answer questions #1** (10 points)

   (a) What class or classes can access and use `public` member variables and methods? What class or classes can access and use `private` member variables and methods? (2 points)

   (b) **Why** declare a `member variable private`? (2 points)

   (c) What technique or mechanism can you use to **give controlled access to a private member variable**? That is, if you have a private member variable in your class, how can a programmer using the class read the private member variable and/or change it? (2 points)

   (d) Name 3 ways that declaring and using a `constructor` is **different** from declaring or using a `regular method`. What is a constructor used for? (2 points)

   (e) What is `null`? When does it come up / when do we see it produced? (2 points)
2. **Short answer questions #2** (10 points)

(a) Describe the **rules** governing what you can do within a **static method** – specifically, its access to instance or static variables, and the ability to call instance or static methods. Briefly describe why. (2 points)

(b) Which is faster, **binary search** or **linear search**? Why...and under what conditions? Are there any assumptions that are made about the Array being searched? (2 points)

(c) What's the difference between a **class** and an **object**? (2 points)

(d) What is the **StringBuilder** class? How is it different from using a regular **String**? (2 points)

(e) What does the keyword, **this**, signify? When should it be used? (2 points)
3. Read the code in the left column. Write the output of the code in the right column. **Compile error, nothing and the actual output are all valid answers.** (10 points)

<table>
<thead>
<tr>
<th>Code (syso is short for System.out.println())</th>
<th>Output (compile error, nothing and actual output are valid)</th>
</tr>
</thead>
<tbody>
<tr>
<td>// imagine you have a stack of ints that // supports push, pop, peek and empty. What is // the output of the code below?</td>
<td></td>
</tr>
<tr>
<td>syso(stack.empty());</td>
<td></td>
</tr>
<tr>
<td>stack.push(4);</td>
<td></td>
</tr>
<tr>
<td>stack.push(2);</td>
<td></td>
</tr>
<tr>
<td>stack.push(12);</td>
<td></td>
</tr>
<tr>
<td>stack.push(5);</td>
<td></td>
</tr>
<tr>
<td>syso(stack.peek());</td>
<td></td>
</tr>
<tr>
<td>syso(stack.pop());</td>
<td></td>
</tr>
<tr>
<td>stack.pop();</td>
<td></td>
</tr>
<tr>
<td>stack.pop();</td>
<td></td>
</tr>
<tr>
<td>syso(stack.peek());</td>
<td></td>
</tr>
</tbody>
</table>

| public class Foo { | |
| public static void main(String[] args) { | |
| Foo f1 = new Foo(4); | |
| Foo f2 = new Foo(); | |
| System.out.println(f1.n); | |
| System.out.println(f2.n); | |
| f1 = f2; | |
| addFive(f1); | |
| System.out.println(f1.n); | |
| System.out.println(f2.n); | |
| } | |
| public static void addFive(Foo f) { | |
| f.n += 5; | |
| } | |
| int n; | |
| Foo() {} | |
| public Foo(int n) { | |
| this.n = n; | |
| } | |
Find 5 of the 6 errors in the code below. They can be compile time, run time or logical errors.

```java
import java.lang.reflect.Array;
import java.util.Arrays;

public class Stacky {
    private int[] elements;
    private int size;

    public Stacky(int capacity) {
        elements = new double[capacity];
        // this controls the size of the stack... so it should
        // *always* accurately determine where the end of the stack is
        size = 0;
    }

    public void push(int x) {
        if (size >= elements.length) {
            int[] newElements = new int[size * 2];
            for(int i = 0; i < newElements.length; i++) {
                elements[i] = newElements[i];
            }
            elements = newElements;
            elements[size - 1] = x;
        }
    }

    public int pop() {
        size += 1;
        return elements[size];
    }

    public static int getSize() {
        return this.size;
    }
}
```

(a) _______________________________________________________
(b) _______________________________________________________
(c) _______________________________________________________
(d) _______________________________________________________
(e) _______________________________________________________

5. True or False: (10 points)

(a) _____ You must declare a visibility modifier (public, private or protected) before member variables and methods. Without a modifier, you will get a compiler error.

(b) _____ A static method can be invoked both from the class name as well as from an instance:

```java
ClassName.myStaticMethod();
ClassName instance = new ClassName();
instance.myStaticMethod();
```

(c) _____ In the following code, the variables greeting1 and greeting2 both have a reference that points to the same copy of "I am a string" in memory.

```java
Greeting1 = "I am a string";
Greeting2 = new String("I am a string");
```

(d) _____ In a program that uses Processing, the setup() method is called exactly once, and the draw() method is called repeatedly after the setup method.

(e) _____ Is the output of the code below true or false?

```java
String s1 = "abc";
String s2 = "adb";
System.out.println(s1.compareTo(s2) < 0);
```

(f) _____ If a local int variable in a method is not explicitly initialized within that method, it receives a default value of 0.

(g) _____ Binary search requires an Array to be presorted.

(h) _____ Assuming that the Foo class is defined and contains a public member variable, n, both variables, foo1 and foo2 are instances of the Foo class, and the constructor takes the argument passed in and sets the member variable, n... the code below will print out:

```java
// 25
// 12
Foo foo1 = new Foo(12);
Foo foo2 = foo1;
foo1.n = 25;
System.out.println(foo1.n);
System.out.println(foo2.n);
```

(i) _____ If a single constructor is defined in a class, it will have that constructor, as well as the default, no argument constructor available for use.

(j) _____ (new String("hello")) == (new String("hello"))
6. **Two Dimensional Arrays** (20 points) - Write the following two methods that work on 2D Arrays...

(a) // takes a two dimensional Array, and reverses the columns in the original
   // Array passed in (does not return a new Array!)
   public static void reverseColumnsInPlace(int[][] arr)

Example:

<table>
<thead>
<tr>
<th>Initial declaration of m</th>
<th>Calling method with m</th>
<th>m is now...</th>
</tr>
</thead>
</table>
| int[][] m = {            | reverseColumnsInPlace(m); // m looks like:
  {1, 2, 3, 4},          |                         |
  {2, 3, 4, 5},          | {3, 4, 5, 6},           |
  {3, 4, 5, 6}           | {2, 3, 4, 5},           |
}                         | {1, 2, 3, 4}            |

(b) // give back the row with the largest sum; it's ok to return a reference
   // rather than a new Array
   System.out.println(Arrays.toString(getRowWithMaxSum(m)));)

Example:

<table>
<thead>
<tr>
<th>Initial declaration of m</th>
<th>Calling method with m</th>
<th>Value returned is</th>
</tr>
</thead>
<tbody>
<tr>
<td>int[][] m = {</td>
<td>getRowWithMaxSum(m);</td>
<td>an Array is</td>
</tr>
</tbody>
</table>
  {1, 2, 3, 4},           |                        | returned         |
  {2, 3, 4, 5},           | {3, 4, 5, 6},          |
  {3, 4, 5, 6}            |}                      |
7. **Sorting Magical Sticks** (20 points)

(a) Finish the missing implementation of bubble sort method below

(b) Create a class MagicalStick, based on the usage of the main method below

```java
public class Sort MagicalSticks {
    public static void main(String[] args) {
        MagicalStick[] magicStuff = new MagicalStick[4];
        magicStuff[0] = new MagicalStick(7, "wiggle ears");
        magicStuff[1] = new MagicalStick(10, "levitate");
        magicStuff[2] = new MagicalStick(8, "turn into panda");
        magicStuff[3] = new MagicalStick(9, "lazer beamz");
        bubbleSort(magicStuff);
        System.out.println("Sorted by length...");
        for(MagicalStick s: magicStuff) {
            System.out.println(s.getLength() + " " + s.getPower());
        }
    }
    public static void bubbleSort(MagicalStick[] sticks) {
        // implement bubblesort according to the pseudocode below:
        // keep track of swaps
        // while no swaps have happened
        // assume no swaps
        // for every index, up to second to last
        // compare element at index with next element to sort by length
        // if element at index is greater
        // then swap elements
        // keep track of the fact that a swap happened
    }
}
// implement that class that would be in MagicalStick.java
```
8. **My Very Own String Class** (20 points) - Create your own version of a String class called MyString. Implement the following methods and constructors (if necessary, see the unicode character chart on the last page):

```java
// the constructor, initializes object with the char[] Array supplied
MyString(char[] chars)

// returns the character at the specified index
public char charAt(int index)

// give back the number of characters
public int length()

// return a new MyString object that made from a substring of the original, // starting at the index, begin (inclusive), going up to, but not including // the index, end (exclusive)
public MyString substring(int begin, int end)

// gives back a new MyString object that is all lowercase (only letters are // affected)... when adding an int to a char, you'll have to cast the result // to char if you're assigning the result to a char variable
public MyString toLowerCase()

//Example Usage
char[] c = {'v', 'o', 'w', 'e', 'l'};
MyString s = new MyString(c);
System.out.println(s.charAt(0)); // v
System.out.println(s.length());  // 5
System.out.println(s.substring(1, 4)); // MyString object with 'o' 'w' 'e'
```
9. **Triangle Maker** (20 points)

(a) Write a program that asks for the coordinates of 3 points, and prints out:
   i. the perimeter of the resulting triangle (the sum of all sides)
   ii. whether or not the resulting triangle is isosceles (two sides are equal)
   iii. example interaction below:

   Enter point 1 as x,y
   > 0,0
   Enter point 2 as x,y
   > 4,10
   Enter point 3 as x,y
   > 8,0
   The triangle's perimeter is 29.540659228538015
   The triangle is isosceles.

(b) Requirements
   i. The input must be in the format x,y (hint: there's a method that helps you extract x and y)
   ii. Expect that the inputs are integers (hint: Integer.parseInt may come in handy)
   iii. You must create 3 classes:
   iv. **Triangle** - A class that represents a triangle; it should contain 3 Point objects in an Array, and the class should be fully encapsulated (points should not be available by direct data field access)... as well as a couple of methods for finding the perimeter and determining if a triangle is isosceles
   v. **Point** - A class that represents a point in a 2d plane; it's ok to make the x and y value accessible directly. A method should exist in this object that calculates distance
   vi. **TriangleMaker** - the entry point into your program, it should be responsible for all input and output
<table>
<thead>
<tr>
<th>Char Dec</th>
<th>Char Dec</th>
<th>Char Dec</th>
<th>Char Dec</th>
</tr>
</thead>
<tbody>
<tr>
<td>(nul) 0</td>
<td>(sp) 32</td>
<td>@ 64</td>
<td>` 96</td>
</tr>
<tr>
<td>(soh) 1</td>
<td>! 33</td>
<td>A 65</td>
<td>a 97</td>
</tr>
<tr>
<td>(stx) 2</td>
<td># 35</td>
<td>B 66</td>
<td>b 98</td>
</tr>
<tr>
<td>(etx) 3</td>
<td>$ 36</td>
<td>C 67</td>
<td>c 99</td>
</tr>
<tr>
<td>(eng) 4</td>
<td>% 37</td>
<td>D 68</td>
<td>d 100</td>
</tr>
<tr>
<td>(ack) 5</td>
<td>&amp; 38</td>
<td>E 69</td>
<td>e 101</td>
</tr>
<tr>
<td>(bel) 6</td>
<td>' 39</td>
<td>F 70</td>
<td>f 102</td>
</tr>
<tr>
<td>(bs) 8</td>
<td>( 40</td>
<td>G 71</td>
<td>g 103</td>
</tr>
<tr>
<td>(ht) 9</td>
<td>) 41</td>
<td>H 72</td>
<td>h 104</td>
</tr>
<tr>
<td>(nl) 10</td>
<td>* 42</td>
<td>I 73</td>
<td>i 105</td>
</tr>
<tr>
<td>(vt) 11</td>
<td>+ 43</td>
<td>J 74</td>
<td>j 106</td>
</tr>
<tr>
<td>(np) 12</td>
<td>, 44</td>
<td>K 75</td>
<td>k 107</td>
</tr>
<tr>
<td>(cr) 13</td>
<td>- 45</td>
<td>L 76</td>
<td>l 108</td>
</tr>
<tr>
<td>(so) 14</td>
<td>. 46</td>
<td>M 77</td>
<td>m 109</td>
</tr>
<tr>
<td>(si) 15</td>
<td>/ 47</td>
<td>N 78</td>
<td>n 110</td>
</tr>
<tr>
<td>(dle) 16</td>
<td>0 48</td>
<td>O 79</td>
<td>o 111</td>
</tr>
<tr>
<td>(dc1) 17</td>
<td>1 49</td>
<td>P 80</td>
<td>p 112</td>
</tr>
<tr>
<td>(dc2) 18</td>
<td>2 50</td>
<td>Q 81</td>
<td>q 113</td>
</tr>
<tr>
<td>(dc3) 19</td>
<td>3 51</td>
<td>R 82</td>
<td>r 114</td>
</tr>
<tr>
<td>(dc4) 20</td>
<td>4 52</td>
<td>S 83</td>
<td>s 115</td>
</tr>
<tr>
<td>(nak) 21</td>
<td>5 53</td>
<td>T 84</td>
<td>t 116</td>
</tr>
<tr>
<td>(syn) 22</td>
<td>6 54</td>
<td>U 85</td>
<td>u 117</td>
</tr>
<tr>
<td>(etb) 23</td>
<td>7 55</td>
<td>V 86</td>
<td>v 118</td>
</tr>
<tr>
<td>(em) 24</td>
<td>? 56</td>
<td>W 87</td>
<td>w 119</td>
</tr>
<tr>
<td>(can) 25</td>
<td>9 57</td>
<td>X 88</td>
<td>x 120</td>
</tr>
<tr>
<td>(sub) 26</td>
<td>: 58</td>
<td>Y 89</td>
<td>y 121</td>
</tr>
<tr>
<td>(esc) 27</td>
<td>; 59</td>
<td>Z 90</td>
<td>z 122</td>
</tr>
<tr>
<td>(fs) 28</td>
<td>&lt; 60</td>
<td></td>
<td>91</td>
</tr>
<tr>
<td>(gs) 29</td>
<td>= 61</td>
<td>} 93</td>
<td>} 125</td>
</tr>
<tr>
<td>(rs) 30</td>
<td>&gt; 62</td>
<td>^ 94</td>
<td>~ 126</td>
</tr>
<tr>
<td>(us) 31</td>
<td>? 63</td>
<td>_ 95</td>
<td>(del) 127</td>
</tr>
</tbody>
</table>