1) Fill in the Blanks (16 points – 2 points each). Provide the MOST appropriate answers. If choices are given in square brackets, select one of the choices.
   a) The _______selectionsort_________ (give the name) algorithm that we discussed in lecture sorts items by finding the smallest item and putting it into location 0, then finding the next smallest item and putting it into location 1, etc.
   b) The ____physical__________ size of an ArrayList is the length of the underlying array and the _____logical__________ size of an ArrayList is the number of items being stored in the array.
   c) If I binary search an array with N items in it, in the worst case I will need approximately ______lgN_________ (give a function of N) guesses.
   d) If a copy constructor for a class makes a bitwise copy of all of the instance variables, it is making a [shallow, deep] _____shallow_________ copy.
   e) In Java 1.5 autoboxing was introduced. This implicitly converts back and forth between primitive types and their corresponding _______wrapper_________ classes.
   f) Every Java object has a pseudo-instance variable called ________this____________ which is a reference back to that same object (i.e. a self-reference).
   g) A subclass can directly access ________public and protected only________________ [all, public and protected only, public only] instance variables declared in its superclass.
   h) Composition gives a ______has a________ [is a, has a] relationship between the new and old classes while inheritance gives a ______is a________ [is a, has a] relationship between the new and old classes.

2) True / False (12 points – 2 points each). Correct false statements for full credit.
   a) The declaration: String [] A = new String[10]; will create 10 String objects. False – it will create an array of 10 null references.
   b) Two-D Java arrays must be rectangular (i.e. each row must have the same number of columns). False – Java arrays can be ragged.
   c) The Integer.parseInt() method is used in Java convert an int value into its corresponding String. False – it converts the String into an int.
   d) Consider classes Foo and SubFoo, where SubFoo is a subclass of Foo. The Java statement:
      Foo F = new SubFoo(); is legal True
   e) Consider classes Foo and SubFoo, where SubFoo is a subclass of Foo. The Java statement:
      SubFoo S = new Foo(); is legal False -- the "is a" relationship is one way only.
   f) Subclassing polymorphism utilizes method overloading and dynamic binding. False – it utilizes method overriding and dynamic binding
3) **Trace (12 points – 10 + 2).** Give all output, in the correct order and format, produced by execution of the following Java program. **Clearly indicate your output by drawing a box around it. Also state the algorithm that is demonstrated by the trace.**

```java
public class Trace1A
{
    public static int doSomething(int [] A, int val)
    {
        boolean done = false;
        int ans = -1;
        int loc1 = 0;
        int loc2 = A.length - 1;
        while (!done && loc1 <= loc2)
        {
            int loc3 = (loc1 + loc2) / 2;
            System.out.print("Loc1: "+loc1+"  Loc2: "+loc2+"  Loc3: ");
            System.out.println(loc3 + "  Data: " + A[loc3]);
            if (val == A[loc3])
            {
                ans = loc3;
                done = true;
            }
            else if (val < A[loc3])
                loc2 = loc3 - 1;
            else
                loc1 = loc3 + 1;
        }
        return ans;
    }

    public static void runIt(int [] A, int val)
    {
        int check = doSomething(A, val);
        if (check >= 0)
            System.out.println("Loc "+check+" has "+val);
        else
            System.out.println("Other result for "+val);
    }

    public static void main(String [] args)
    {
        int [] D = {10, 15, 20, 25, 30, 35, 40, 45, 50};
        runIt(D, 25);
        runIt(D, 41);
    }
}
```

**Output:**

```
Loc1: 0  Loc2: 8  Loc3: 4  Data: 30
Loc1: 0  Loc2: 3  Loc3: 1  Data: 15
Loc1: 2  Loc2: 3  Loc3: 2  Data: 20
Loc1: 3  Loc2: 3  Loc3: 3  Data: 25
Loc 3 has 25
Loc1: 0  Loc2: 8  Loc3: 4  Data: 30
Loc1: 5  Loc2: 8  Loc3: 6  Data: 40
Loc1: 7  Loc2: 8  Loc3: 7  Data: 45
Other result for 41
```

**The algorithm demonstrated is Binary Search**
4) **Trace (10 points).** Consider the `DataType` class on this page and the Trace2A program on the next page. Give all output, in the correct order and format, produced by execution of the Trace2A program. **Clearly indicate your output by drawing a box around it.**

```java
public class DataType {
    StringBuilder[] A;
    int N;

    public DataType(int val) {
        A = new StringBuilder[val];
        N = 0;
    }

    public DataType(DataType old) {
        A = new StringBuilder[old.A.length];
        for (int i = 0; i < A.length; i++)
            A[i] = old.A[i];
        N = old.N;
    }

    public boolean add(StringBuilder sb) {
        if (N < A.length) {
            A[N] = sb;
            N++;
            return true;
        } else return false;
    }

    public StringBuilder get(int i) {
        if (i >= 0 && i < N)
            return A[i];
        else
            return null;
    }

    public int size() {
        return N;
    }
}
```
public class Trace2A
{
    public static String [] info = {"Harry", "Ron", "Hermione"};

    public static void printIt(DataType D)
    {
        System.out.print("Contents: ");
        for (int i = 0; i < D.size(); i++)
            System.out.print(D.get(i).toString() + " ");
        System.out.println();
    }

    public static void main(String [] args)
    {
        DataType D1, D2;
        D1 = new DataType(5);
        for (int i = 0; i < info.length; i++)
            D1.add(new StringBuilder(info[i]));
        printIt(D1);

        D2 = new DataType(D1);
        D1.get(0).append("Potter");
        D1.add(new StringBuilder("Luna"));
        D2.get(1).append("Weasley");
        D2.add(new StringBuilder("Hagrid"));

        printIt(D1);
        printIt(D2);
    }
}

Answer Comment: The copy constructor for DataType makes neither a shallow nor a deep copy. A new array is made but the StringBuilders within the array are not copied. Therefore, when the object for D2 is made, its array will be sharing the same StringBuilder objects (containing "Harry", "Ron" and "Hermione") that are stored for D1. Thus, appending to any of them will affect both D1 and D2.

Contents: Harry Ron Hermione
Contents: HarryPotter RonWeasley Hermione Luna
Contents: HarryPotter RonWeasley Hermione Hagrid