Quiz 1 — Data Struct. & More (T. I/21–22)

Directions:

- This exam is "paper-based." Answer all the questions in the on-screen editor provided.
- No consultation with other people, notes, books, nor the Internet is permitted. Do **not** use an IDE or run Java code.
- Do **not** leave the full-screen mode. You can chat with the instructors via the built-in chat.
- This quiz is worth a total of 30 points. You have 50 minutes. Good luck!

Problem 1: True/False (5 points)

_T_ 1. When we define a method **static double** sqrt(**double** x) inside a class A, it can be called without having to instantiate the class A.

_F_ 2. Java does *not* allow constructor overloading.

_T_ 3. **int**[] x=**new** **int**[]{1, 3, 5}; correctly creates and initializes an array called x.

_T_ 4. **int**[] y={1, 3, 5}; correctly creates and initializes an array called y.

_F_ 5. The code snippet below will print out [1, 3]

```
int[] a = new int[3];
a[0] = 1;
a[1] = 3;
System.out.println(a);
```

Problem 2: What Will Java Do (10 points)

Carefully consider the following Unknown class.

```
public class Unknown {
    public static int w;
    private static int x;
    private int y;
    public int[] z = new int[5];

    public Unknown() {w += 1; y = w;}

    public static void setX(int i) {x += i;}

    public void setZ(int i) {z[y] = i; y += 1;}

    public int getW() {return w;}
    public int getX() {return x;}
    public int getY() {return y;}
    public int getZ(int i) {return z[i];}
}
```

Each program below is a complete implementation that refers to the class Unknown (above). For each program, write down the output for each print statement, or if that line results in an error, write "ERROR" and a brief explanation.

1.
```
import java.util.Arrays;
public class P1 {
  public static void main(String[] args) {
      System.out.println(Unknown.w);      // ______0______ 1
      System.out.println(Unknown.x);      // ______0______ 2
      System.out.println(Unknown.y);      // non-static variable 3
      System.out.println(Arrays.toString(z)); // can't be referenced 4
                                               from a static context
  }
}
```

```
2. public class P2 {
     public static void main(String[] args) {
         Unknown u1 = new Unknown();  W=1  y=1
         Unknown.setX(5);  x=5
         System.out.println(u1.getY()); // __1__________________ 5

         Unknown u2 = new Unknown();  W=2  y=2
         Unknown.setX(5);   x=10
         System.out.println(u2.getW()); // ____2________________ 6
         System.out.println(u2.getX()); // ____10_______________ 7
         System.out.println(u2.getY()); // ____2________________ 8
     }
   }
```

```
3. public class P3 {
     public static void main(String[] args) {
         Unknown u1 = new Unknown();  W=1  y=1
         u1.setZ(5);  {0,5,0,0,0}   y=2
         System.out.println(u1.getZ(2)); // ____0________________ 9

         Unknown u2 = new Unknown();   W = 2, y = 2
         u2.setZ(4);   {0,0,4,0,0}
         System.out.println(u2.getZ(2)); // ____4________________ 10
     }
   }
```

Problem 3: Fill in the Blanks (5 points)

The following snippet is a method inside a class. The method takes as input an array of `String`s and returns an array of **int**s storing the lengths of the strings in the input array. For example, abusing the array notation for brevity, expect `len({"h", "el", "l", "o,W", "orld!"})` to return `{1,2,1,3,5}`. It is possible that the output is an empty array (i.e., array of size 0). Complete the code below by filling in the blanks.

```
public int[] len(_String[]___ 1 arr) {
    _int[]_ 2 lengths = new int[arr.length];

    for (_int i=0_____ 3; _i<arr.length_ 4; i++) {
        lengths[i] = _arr[i].length()________ 5;
    }

    return lengths;
}
```

Problem 4: Singly-Linked List With a Sentinel (2 + 2 + 6 points)

The class `SLList` below implements an **int** singly-linked list *with* a front sentinel, like was discussed in class. More precisely, the instance variable `sen` refers to the sentinel node and thus `sen.rest` refers to the real first node. For this problem, you'll add 3 extra methods to the `SLList` class:

(1) a method **public boolean** `isEmpty()` that returns whether the list is empty. The list is considered empty if it has no real elements (other than the sentinel).

(2) a method **public void** `addFirst(int x)` that adds a new entry `x` at the start of the list (i.e., after the sentinel).

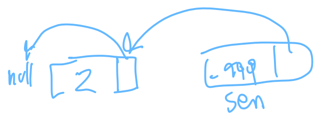
(3) a method **public double** `getAverage()` that returns the average of the numbers in the list. Remember that the sentinel isn't part of the list, and the average is the sum of the numbers divided by the length of the list. If the list is empty, the method should return `0.0`.

(You cannot add new member variables to either class, nor can you modify existing methods/constructors. Your methods can be iterative or recursive, or any combination of them that works.)

```
public class SLList {
    private static class IntNode {
        int head; // an int data item
        IntNode rest; // ref to the next node

        public IntNode(int val, IntNode r) {
            this.head = val; this.rest = r;
        }
    }

    private IntNode sen;

    public SLList() { sen = new IntNode(0, null); }

    public int getFirst() {
        return sen.rest.head;
    }
}
```

```
public boolean isEmpty() { return sen.rest == null ;}

public void addFirst( int x ) {
    sen.rest = new IntNode (x, sen.rest);
}

public double getAverage() {
    if (this.isEmpty()) {return 0.0;}
    else {
        IntNode p = sen.rest ;
        double sum = p.head ;
        p = p.rest ;
        while (p != null) { sum+= p.head ; p= p.rest ;}
        int count = 0;
        IntNode q = sen.rest;
        while ( q != null) { count +=1 ; q = q.rest}
        return sum/count ;
    }
}
```

null
-999
sen



(3) a method **public double** getAverage() that returns the average of the numbers in the list. Remember that the sentinel isn't part of the list, and the average is the sum of the numbers divided by the length of the list. If the list is empty, the method should return 0.0.

(You cannot add new member variables to either class, nor can you modify existing methods/constructors. Your methods can be iterative or recursive, or any combination of them that works.)

```java
public class SLList {
    private static class IntNode {
        int head; // an int data item
        IntNode rest; // ref to the next node

        public IntNode(int val, IntNode r) {
            this.head = val; this.rest = r;
        }
    }

    private IntNode sen;

    public SLList() { sen = new IntNode(0, null); }

    public int getFirst() {
        return sen.rest.head;
    }
}
```