1. (2 pts each, 8 pts total) What is the value of either ans or result after each of the statements below. Assume the declarations:

\[
\begin{align*}
\text{double } \text{ans} &= 2.0; \\
\text{int } \text{result} &= 1;
\end{align*}
\]

\begin{align*}
a. \text{ result } &= 11 \% 4; \\
b. \text{ result } &= 4 \% 10; \\
c. \text{ result } &= 10 / 4; \\
d. \text{ ans } &= 3 / 10 + 7.5;
\end{align*}

2. (2 pts each, 6 pts total) What is the value (true or false) of each of the following Boolean expressions assuming

\[
\text{int } \text{x} = 10;
\]

\begin{align*}
a) \ x \neq -1 \ || \ x \ == \ 10 & \quad \text{True or False or Incorrect Syntax} \\
b) \ !(\ 20 < x < 70) & \quad \text{True or False or Incorrect Syntax} \\
c) \ !(x == 4 \ &\ & x > 5) & \quad \text{True or False or Incorrect Syntax}
\end{align*}
3. (3 pts each, 9 total) What is the value of the following code snippets?
   a. Snippet:

   ```java
   int sum = 0;
   int n = 1;
   while (n < 7)
   {   
       sum = sum + n;
       n = n + 2;
   }   
   System.out.println("sum = " + sum);
   ```
   Output: ____________________________

   b. Snippet:

   ```java
   int cnt = 0;
   for (int j = 0; j < 50; j++) {
       for (int i = 0; i < 10; i++) {
           cnt++;
       }
   }
   System.out.println("cnt = " + cnt);
   ```
   Output: ____________________________

   c. Snippet:

   ```java
   String greet = "Happy New Year!";
   String greet2 = greet.substring(1,8);
   System.out.println(greet2);
   ```
   Output: ____________________________

4. (6 pts total) 1D Arrays:
   a. (2 pts) Write code that declares and creates a 1-dimensional array of doubles called dNums. The length of the array is 10.

   ```java
   double[] dNums = new double[10];
   ```

   b. (4 pts) Write a for-loop which loads the array with random numbers in the range 0 to 1.
5. (18 pts) For each variable name listed at top, indicate a "D" on the line where the variable is declared. Mark an x in boxes to indicate scope. Circle "P" at the top if the variable is formal parameter, "M" if it is a member variable and "L" if it is a local variable.

|   | i | j | k | n | c | a | b | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
6. (12 pts total) 2-Dimensional Arrays: Suppose `myNums` is a 2-dimensional array of integers with 5 rows and 8 columns.
   a. (2 pts each, 4 pts total) What is the value of each of the following
      
      ```
      myNums.length  
      myNums[2].length
      ```

   b. (2 pts each, 8 pts total) What is the type of each of the following where the type should be selected from the list below:
      A. 1-D array of doubles
      B. 1-D array of ints
      C. 2-D array of Strings
      D. 2-D array of ints
      E. ArrayList of Strings
      F. ArrayList of ints
      G. int
      H. String
      I. double
      
      **Circle the answer:**
      
      ```
      myNums[0]     A B C D E F G H I
      myNums.length A B C D E F G H I
      myNums        A B C D E F G H I
      ```

7. (18 pts) Classes:
   a. (15 pts) Create a Vehicle class consisting of the following:
      - Two private instance variables for the vehicle’s type (String) and year (int).
      - A constructor which sets the value of both instance variables.
      - A setter & getter for year.
      - A toString method
      - The code should be consistent with the following declaration and print statement:

      ```java
      Vehicle vehicle = new Vehicle("truck", 2010);
      System.out.println(vehicle);  
      ```
      
      Which should output:  **Type: truck, Year: 2010**
public class Vehicle {

   // Private instance member variables for type and year:

   // Constructor:

   // Getter and Setter for year

   // toString

}

b. (3 pts) Suppose you want to create a subclass of Vehicle called Car which has an additional private instance String variable called make (e.g. whose value could be "Ford"). Below, write only the constructor for Car. It should be consistent with the declaration:

   Car car = new Car("car", 2010, "Ford");

   // Constructor:
8. (1 pt each, 23 pts total) **True and False**: Please circle T or F

2.1) T or F: Object parameters are passed by reference.

2.2) T or F: The keyword word `static` is used to indicate instance methods and variables.

2.3) T or F: A class can implement more than one interface.

2.4) T or F: A class cannot have more than one superclass.

2.5) T or F: In an abstract class, not all of the methods are implemented.

2.6) T or F: If one changes the value of a class variable, the value is **not** changed for all objects of that type.

2.7) T or F: If Car is a subclass of Vehicle, then it is ok to have the declaration:

```
Car c = new Vehicle();
```

2.8) T or F: An object may be created from an abstract or concrete class.

2.9) T or F: The keyword `super` is used to call the constructor of a superclass.

2.10) T or F: To create a subclass, one uses the `implements` keyword.

2.11) T or F: A subclass has access to protected instance variables of its superclass.

2.12) T or F: In the RGB color format, c represents black if

```
Color c = new Color(255,255,255);
```

2.13) T or F: When a user presses a button, an event is generated.

2.14) T or F: A superclass inherits data and behavior from a subclass.

2.15) T or F: GUI components such as JButtons can be found in the Swing library.

2.16) T or F: It is never ok for two methods in a class to have the same name.

2.17) T or F: An anonymous inner class can access to the member variables of the enclosing class.

2.18) T or F: The `catch` part of a try-catch is used to indicate what to do if no errors are generated.

2.19) T or F: A constructor should always have a `void return value`.

2.20) T or F: An class's constructor is called when the keyword `new` is used.

2.21) T or F: Once an object is garbage collected, it can still be retrieved if needed again.

2.22) T or F: Private *member variables* can be accessed outside of the class by using setters and getters.

2.23) T or F: Suppose `setValue` is a method with one parameter of type int. When **calling** the method, you need to provide an **actual parameter**, e.g. `setValue(3)`. And when **declaring** the method, you need to provide a **formal parameter**, e.g., `setValue(int n)`. 