**Q1:** ***(Invoice Class)*** Create a class called Invoice that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance variables—a **part number** **(type String),** a **part description (type String),** a **quantity of the item being purchased (type int)** and a **price per item (doubl**e). Your class should have a constructor that initializes the four instance variables. Provide a ***set* and a *get***method for each instance variable.

In addition, provide a method named *getInvoiceAmount()* that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test application named **InvoiceTest.java** that demonstrates class Invoice’s capabilities.

**Q2: *(Date Class)*** Create a class called **Date** that includes three instance variables—**a month (type int), a day (type int) and a year (type int)**. Provide a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a ***set and a get***method for each instance

variable. Provide a method ***displayDate*** that displays the month, day and year separated by forward slashes (/). Write a test application named **DateTest.java** that demonstrates class Date’s capabilities.

**Q3: *(Sides of a Right Triangle)*** write an application that reads three nonzero integers from a file and determines and prints whether they could represent the sides of a right triangle.

**Q4: *(Calculating Sales)*** an online retailer sells five products whose retail prices are as follows: Product 1, $2.98; product 2, $4.50; product 3, $9.98; product 4, $4.49 and product 5, $6.87. Write an application that reads a series of pairs of numbers as follows:

a) Product number

b) Quantity sold

Your program should use a switch statement to determine the retail price for each product. It should calculate and display the total retail value of all products sold. Use a sentinel-controlled loop to determine when the program should stop looping and display the final results.

(Sentinel-controlled loop: **Sentinel**-**controlled** repetition is sometimes called indefinite repetition because it is not known in advance how many times the **loop** will be executed. It is a repetition procedure for solving a problem by using a **sentinel** value (also called a signal value, a dummy value or a flag value) to indicate "end of data entry".

Q5: ***(Fibonacci Series)*** the Fibonacci series

0, 1, 1, 2, 3, 5, 8, 13, 21…

Begins with the terms 0 and 1 and has the property that each succeeding term is the sum of the two preceding terms.

1. Write a method ***fibonacci( n )*** that calculates the *n*th Fibonacci number. Incorporate this method into an application that enables the user to enter the value of n.

**Q6: *(Using the Enhanced for Statement)***

Write an application that uses an enhanced for statement to sum the double values passed by the command-line arguments.

**Q7: (ArrayList switchPairs)** write a method *switchPairs()* that switches the order of values in an **ArrayList of Strings** in a pairwise fashion. Your method should switch the order of the first two values, then switch the order of the next two, switch the order of the next two, and so on.

For example, if the list initially stores these values: {"four", "score", "and", "seven", "years", "ago"} your method should switch the first pair, "four", "score", the second pair, "and", "seven", and the third pair, "years", "ago", to yield this list: {"score", "four", "seven", "and", "ago", "years"}.

If there are an odd number of values in the list, the final element is not moved. For example, if the original list had been: {"to", "be", "or", "not", "to", "be", "hamlet"} It would again switch pairs of values, but the final value, "hamlet" would not be moved, yielding this list: {"be", "to", "not", "or", "be", "to", "hamlet"}

**Q8: (Array List remove Bad Pairs)**

Write a method ***removeBadPairs()*** that accepts an ArrayList of integers and removes any adjacent pair of integers in the list if the left element of the pair is larger than the right element of the pair. Every pair's left element is an even-numbered index in the list, and every pair's right element is an odd index in the list.

For example, suppose a variable called list stores the following element values: [3, 7, 9, 2, 5, 5, 8, 5, 6, 3, 4, 7, 3, 1] We can think of this list as a sequence of pairs: (3, 7), (9, 2), (5, 5), (8, 5), (6, 3), (4, 7), (3, 1). The pairs (9, 2), (8, 5), (6, 3), and (3, 1) are "bad" because the left element is larger than the right one, so these pairs should be removed. So the call of ***removeBadPairs(list);*** would change the list to store the following element values: [3, 7, 5, 5, 4, 7].

If the list has an odd length, the last element is not part of a pair and is also considered "bad;" it should therefore be removed by your method. If an empty list is passed in, the list should still be empty at the end of the call. You may assume that the list passed is not null. You may not use any other arrays, lists, or other data structures to help you solve this problem, though you can create as many simple variables as you like.

**Q9: (String practice)**

Write a program which use the following string results

String results = "Manchester United 1 Chelsea 0, Arsenal 1 Manchester United 1, Manchester United 3 Fulham 1, Liverpool 2 Manchester United 1, Swansea 2 Manchester United 4";

And outputs how many wins Manchester United had, how many games they drew, and how many Manchester United lost.

Extend the program to work out how many goals Manchester United scored and how many they conceded.

Suppose a win gains you 3 points, a draw 1 point, and a loss no points. Have your program work out how many points in total Manchester United have acquired.

When you have finished this exercise, the output in your textbox, listbox or console should be as follows:

