Montgomery College
2019 Programming Competition
Advanced Teams

Scoring:

- Each problem is worth of 10 points. There are 6 problems.
- Teams are ranked based on the number of points received, the highest score first.
- Teams that tie on points will be ranked according to the number of problems solved/attempted.
- Teams that tie on both points and the number of problems solved/attempted will be ranked according to the tie-break problem #6.

Criteria	**Points Awarded**
Compiles, runs, passes all (100%) testing	**10**
Compiles, runs, passes more than 50% and less than 100% of testing	**7**
Compiles, runs, passes less than 50% of testing	**3**
Compiles, runs, passes all testing, but fails to implement the solution as specified. For example, requirements call for the use of a two-dimensional array, the submitted solution did not use a two-dimensional array. Or, the requirement asks for a recursive implementation, the submitted solution did not use recursion.	**5**
Compiles, runs, hardcodes the expected output	**0**
Does not compile	**0**

Teams are required to return this problem set to the competition staff at the end of the competition.

Problem #1: Vector3D Class

Write a class `Vector3D` that contains three instance variables `x`, `y` and `z` of type `double`. Provide a constructor that accepts the values of `x`, `y` and `z` and initializes the corresponding instance variables with those values. Provide a public method named `length` that computes and returns the length of the vector by computing the distance between the point whose coordinates are `x`, `y` and `z` and the origin. The class `Vector3D` should implement the predefined interface `Comparable` and consequently override the method `compareTo` that compares two vectors according to their lengths by calling the method `length`.

In addition, it should override the `toString` method so it formats a vector as follows `(x, y, z)`.

A second class should contain the main method. It should read in vectors from a file named `vectors.txt` that contains at most 20 vectors. It should determine the vector with the greatest length and display that vector and its length. If the file contains two such vectors it should display the first one.

Given the following input file that contains one vector per line:

```
10.5 7.4 10.4
6.3 5.8 11.8
13.6 9.3 12.7
6.3 8.3 2.9
11.5 12.6 3.9
9.3 12.7 13.6
10.3 2.5 8.3
```

The program should produce the following output:

```
Longest vector is (13.6, 9.3, 12.7) of length 20.802
```

Notes: Use the following formula to calculate the length of the vector:

Square root of $(x^2 + y^2 + z^2)$

Problem #2: Weighted Shape Hierarchy

Write an abstract class `WeightedShape` that contains one private instance integer variable `weight` and one private class (static) integer variable `totalWeight`. It should contain a constructor that is supplied with the weighting value that indicates the importance of the shape and initializes the instance variable `weight` with the value of that parameter and adds that weight to the class variable `totalWeight`. It should also have an abstract method named `perimeter` that returns a `double` value, and a public method `weightedPerimeter` that returns the product of the weight of that shape times its perimeter. In addition it should have a class (static) method that returns the total weight of all shapes that have been created.

It should have two subclasses. The first subclass, `Rectangle`, should have two instance variables that contain the width and height of the rectangle. The second subclass, `Circle`, should have one instance variable that contains the diameter of the circle. Each one should have a constructor that that is supplied with the weight and values necessary to initialize the other instance variables. Each subclass should also override the abstract method `perimeter` computing and returning the rectangle's perimeter in the case of the `Rectangle` class and computing and returning the circle's circumference for the `Circle` class.

A third class should contain the main method, which contains the following array:

```
WeightedShape[] shapes = {new Circle(2, 4.5),
      new Rectangle(1, 1.0, 5.6), new Circle(1, 3.5)};
```

The main method should also contain a for-each loop that iterates through the array to compute the total weighted perimeter of all the shapes and then compute and display the weighted average in 3 decimal places. The output should be as follows:

```
Weighted average perimeter = 13.117
```

Problem #3: Hexadecimal Selection Sort

Write a program that reads in a list of ten integers, written in hexadecimal, base 16, from the keyboard and store them in an array. You may assume that each of these numbers contains no more than three hexadecimal digits and that the alphabetic letters a-f will always appear in lower case.

These integers should be sorted using a selection sort, which segments the array into a sorted section and unsorted section. Initially the whole array is unsorted. The unsorted section is repeatedly searched for the smallest value in the unsorted section. Once found, that smallest value is swapped with the position in the array after the last sorted value, increasing the number of sorted elements by one and decreasing the unsorted section by one. The process is repeated until no elements remain in the unsorted section.

After each additional element is added to the sorted section the contents of the entire array should be displayed on a separate line.

The following is a sample run of this program:

```
Enter ten hexadecimal integers: 7d 1df c99 45 f3 a 1b cc 10 23

00a 1df c99 045 0f3 07d 01b 0cc 010 023
00a 010 c99 045 0f3 07d 01b 0cc 1df 023
00a 010 01b 045 0f3 07d c99 0cc 1df 023
00a 010 01b 023 0f3 07d c99 0cc 1df 045
00a 010 01b 023 045 07d c99 0cc 1df 0f3
00a 010 01b 023 045 07d c99 0cc 1df 0f3
00a 010 01b 023 045 07d 0cc c99 1df 0f3
00a 010 01b 023 045 07d 0cc 0f3 1df c99
00a 010 01b 023 045 07d 0cc 0f3 1df c99
```

Problem #4: Alphabetic Inflection Points

Write a program that reads in a string of lower case letters from the keyboard and determines the alphabetical inflection points that include both peaks and valleys. A peak is the point where the sequence of letters changes from alphabetic order to reverse alphabetic order. A valley is the point where the sequence of letters changes from reverse alphabetic order to alphabetic order.

A check should be made first to ensure that no letter appears twice in a row and that all characters are lower case letters. If either check fails, an error message should be displayed and the program should terminate.

The output should contain each inflection point labeling it as either a peak or valley and displaying the three letters associated with that point of inflection.

The following are three sample runs of this program:

```
Enter a string of lower case letters: abgjgekkgiop
Contains two of the same characters in a row

Enter a string of lower case letters: abcJndr5s
Contains characters that are not lower case letters

Enter a string of lower case letters: abdghfdcbgklprogbdjz
Peak: ghf
Valley: cbg
Peak: pro
Valley: gbd
```

Problem #5: Search for Closest and Furthest Pairs

Write a program that reads in ten integers into an array. If there are any duplicates in the array, display a message and terminate the program. Otherwise search that array for the pair of distinct integers that are numerically closest together and the pair that is numerically furthest apart. If there is more one pair of either kind, choose the pair containing the element that appears first in the list. Finally, display those two pairs.

The following are two sample runs of this program:

```
Enter 10 integers: 34 83 92 12 29 123 16 83 98 16
Array contains duplicates
```

```
Enter 10 integers: 13 95 187 25 141 26 98 24 94 48
Pair numerically closest together is: 95 94
Pair numerically furthest apart is: 13 187
```

Problem #6 (Tie-Breaker): Sort Ragged Array

Write a method called sort7thRaggedArray that takes a 2D ragged array.
The method will return a new sorted ragged array based on the following rule.
The rows should be sorted according to the sum of each element which is divisible by 7 in the ascending order.

The following is a sample run of this program:

```
Given ragged array:
[39, 27, 1, 42, 35, 3],

[14, 93, 91, 90],

[5, 56, 10, 98, 2],

[105, 7, 5, 17]


the sum would be

row1: 77

row2: 105

row3: 154

row4: 112

so, the method would return an 2d array with the following order:

{{39,27,1,42,35,3},
{14,93,91,90},
{105,7,5,17},
{5,56,10,98,2}}
```