OOPS Lab

Programming Assignment № 5

Polymorphism: Advanced Operator Overloading

CSE Department, JMI 12/09/2023

Read carefully before you begin:

- Total Marks: 30. Each question carries 10 marks.
- You have 2 hours to complete the assignment. Failure to have your program evaluated before you leave the lab will cause forfeiture of the grade for that lab.
- In order to receive full marks, you must demo the full working code and show the output and given an explanation of your approach where applicable.
- Please **save your code** throughout the semester in a place where you do not lose it. You will be required to submit it at the end.
- Use proper filenaming conventions and commenting. Code that is hard to read or understand will incur a penalty.
- Collaboration must kept to general discussions only. Please do NOT share code or directly share answers with each other. Plagiarism is unacceptable.
- Note: Your Point class must have all of the previous functionality from earlier lab assignments. If it does not, please spend time after today's lab to catch up. You need to have distFrom, midPoint, all constructors, the static count, the quadrant-wise counts and the k-Nearest neighbours functionality. If you do not have the optional physics simulation, it is all right but I encourage you to try it out.

Problem 1 : Subscript and Stream Operator Overloading (10 marks)

1. Implement a function inside your point class called **l2Norm** that returns the Euclidean distance of a point from the origin.
2. Overload the subscript operator "[]" so that it works both when a Point object is an l-value and when it is an r-value.
3. Overload the stream out operator $<<$ for your Point class. It should function as shown in the listing below.

Listing 1: Sample C++ code – streamt output in a 2D Point C++. (Source: Saif Ali)

```
#include <iostream>

int main()
```

```
{
    Point p1(1,2);
    cout << p1  ;
}

Output:
"Point (1,2), Norm = 5"
```

Problem 2: Insertion Sort (10 marks)

1. Make arrays of Point objects of size n = $\{10, 100, 1000, 10000\}$.
2. Implement the insertion sort algorithm for Point objects using the subscript operator, the less than or greater than operator
3. Print the sorted list of points using the stream out operator

Problem 3: Performance optimization. (10 marks)

1. Measure the execution time of your code for each value of n.
2. Draw a graph by hand of the execution time (y-axis) versus array size (x-axis). How does your code scale as n increases?
3. Write an insertion sort for "int" types and compare the performance. Does object-oriented code incur a performance overhead?

[OPTIONAL] Problem 4: k-Means Clustering

1. Implement the k-Means clustering algorithm for an array of 500 random points.
2. Hint: Read this, https://reasonabledeviations.com/2019/10/02/k-means-in-cpp/