Roll_no : 21BCS046

Name : Bushra Shahzad

System Specifications:

Processor: AMD Ryzen 5 5600H with Radeon Graphics 3.30 GHz

Installed RAM: 16.0 GB (13.9 GB usable)

System type:64-bit operating system, x64-based processor

OS: Windows 11 Version 22H2 Build 22621.2715

AMD64 Family 25 Model 80 Stepping 0

L2CacheSize	3072 KB
L3CacheSize	16384 KB

gcc version 6.3.0 (MinGW.org GCC-6.3.0-1)

Java version "21.0.1" 2023-10-17 LTS

Java(TM) SE Runtime Environment (build 21.0.1+12-LTS-29)

LAB I

Implemented the code in lab.

LAB II

Q1. Line someLine = Line(12.0)

On executing above line of code, only parameterised constructor is called. (copy constructor not called)

Also note that above line of code will only work when there is a copy constructor in the class Line and the argument of which is (const line &l1), the const keyword with the argument object is necessary. This is because an rvalue cannot bind to a non-const reference.

If we make the parameter of the copy constructor a non-const reference, it means we could potentially modify the source object while creating the copy. However, it is generally expected that copy constructors should not modify the source object.

Q2. Why can't we call a copy constructor by value?

When we call a copy constructor by value, it creates a copy of the object that is being called. This can lead to an infinite loop and stack overflow, because the copy constructor will then call itself to make a copy of the argument, and so on.

To avoid this, we must pass the object to the copy constructor by reference. This ensures that the copy constructor only creates one copy of the object, and that the original object is not modified.

Q3. Is destructors overloading possible? If yes then explain and if no then why?

In C++, it's not possible to overload the destructor. A destructor does not require any arguments or return any value and if we try to define multiple destructors with different parameters or with different names, it will result in a compilation error. However, if we wish to do different things when destroying an object depending on certain circumstances, we can use if statements in destructor.

Q4. The constructor is called when an object is instantiated. Suppose your computer hangs before the constructor can fully execute, will the object be created? How will you test this?

If the constructor is unable to finish its execution, the object is not considered valid, and its destructor won't be called.

Q5. What happens when we write only a copy constructor – does the compiler create a default constructor? Demonstrate your answer by code.

Compilation error will pop up if we write only the copy constructor and there is no default constructor. The compiler doesn't create a default constructor if we write any constructor even if it is a copy constructor.

6. What happens when we write a normal constructor and don’t write a copy constructor? Demonstrate.

The compiler will automatically generate a default copy constructor if we don’t write our own. The compiler creates it, even if we have written other constructors in a class.

Lab Assignment II - Additional Question - Copy Constructors

Q1. Add a copy constructor to your Line class and use it to instantiate Line objects using other Line objects.

As already implemented in lab, I instantiated Line objects using other Line objects.

```
//Line obj1(20);
```

```
//Line obj2(obj1);
```

LAB III

Problem 1: Access Modifiers

Q1. Identify the access modifiers in the class declaration.

Ans. In the given class declaration, public is the access modifier used for member functions which means they will be accessible by anyone. However, the Point class has data members declared in private scope which on the contrary implies not available/accessible outside Point class.

Q2. What is the scope of the data attributes?

Ans. Though not explicitly declared, data attributes are privately declared as the convention in C++. Private scope mean they can not be accessed outside class by anyone. One requires getters/setters for the same.

Q3. What is the scope of the member functions?

Ans. All member functions are declared under the public access scope. This means they are accessible from outside the class by object.

Q4. Implement the class defining each of the functions, constructors and destructor.

Ans. Coded

Q5. Assess the truth or falsehood of the following statement by experimentation. Demonstrate using code. The member functions of a class can access the private members of all objects of that class.

Ans. The statement is true. Member functions of a class can access the private members of all objects of that class. In our code we have make use of public member functions to access and manipulate the private members _x and _y of different objects of the Point class.

Write tests to verify or falsify the following claims:

1. Static data members must be defined and initialized outside the class definition.

Ans. Yes.

```
#include <iostream>

class Point {

public:

    static int count;

    //static int check_count=0; // gives error

};

int Point::count = 0; // defining and initializing the static member outside the class
```

2. Memory is allocated for static data members once they are defined even though no objects of that class have been instantiated.

Ans. #include <iostream>

```
using namespace std;

class Point {

public:

     static int count;          // Static data member

};

int Point::count = 100;   defining and initializing the static member outside the class

int main() {

  // Accessing the static data member without creating an object
```

```
    cout << "Count " << Point::count << endl;
    cout<<"Size of Count "<<sizeof(Point::count)<<endl;
    cout<<"Address of Count "<<&(Point::count)<<endl;
// the address is displayed implying memory have been allocated to static member
    // even before any object has been created
      Point p1;      // Now creating objects of the class
return 0;
}
```

3. Static member functions cannot access non-static data members.

Ans. // Test for Static member functions cannot access non-static data members.

```
#include <iostream> using namespace std;
class Point {
private:
    int x;  int y;
public:
    static void staticMemberFunction01() {
        // uncommenting the line below as it result in a compilation error
        // as staticMemberFunction01 is static and cannot access non-static members directly.
        //cout << "X: " << x << endl;
        cout<<"Static member function 1 called!\n";
    }
    static void staticMemberFunction02(Point &p) {
         p.x=78;
         p.y=87;
         cout<<"Static member function 2 with point as parameter called.\n";
    }
};
int main() {
    Point p;
    p.staticMemberFunction01();
    p.staticMemberFunction02(p);

// As noted, if we pass an object as argument to the static member function then it can access that object's data non static data members.

    return 0;

}
```

4. Static const data members can be initialized within the class definition but not outside it.

Static const data members can be initialized within the class definition but not outside it because they are only initialized once. This ensures that they are always initialized correctly and that they can be used safely by all objects of the class.

5. Const member functions cannot change the state of an object, i.e, they cannot change the values of any of the data members.

Ans. In the implementation of setCoords and printCoords, we observed that setCoords is non const while printCoords is const. If in the printCoords we write code to modify _x or _y then it would show error because const member functions cannot change the values of any data members.

LAB IV

Q1. Identify the overloaded operator declaration inside the class definition.

Ans. Addition operator has been overloaded in Point operator + (const Point & p) const.

2. Identify the return type and argument type.

Ans. Return type is Point, argument is (const Point &p) that is a Point object by reference is passed.

3. What do the two const keywords mean? What restrictions do they impose on the functionality of the operator?

Ans.

const Point &p - This specifies that the parameter p is a constant reference to a Point object. It means that within the body of the function, we cannot modify the object referred to by p. It's a way of indicating that the argument p should not be changed within the function.

const after function parameters - The second const specifies that the function cannot modify the state of the object or the class instance (or any member of it) on which it was called.

4. Implement the operator by defining an appropriate function outside the class definition.

Ans. Implemented in code.

5. Test your implementation by adding 2 points. Which one is that "this" object and which

one is "that" object?

Ans. Implemented in code.

6. Chain the operator and try to add more than two points. How many points can you chain?

We can chain more than two points and as much as we want, there is no limit as such. Chaining of multiple points have been implemented in code.

LAB V

Problem 1: Subscript and Stream Operator Overloading

I faced some issues in overloading subscript operator primarily because it should work for both r value and l value. As taught in the class to make it work for both, reference should be returned. But when we are returning a reference to a local variable ans =0.0 (in case of invalid index) then a warning is issued stating reference to local variable ans returned. One way to deal with it is by declaring static double ans variable but then again it will persist even after the function call which is not a recommended practice.

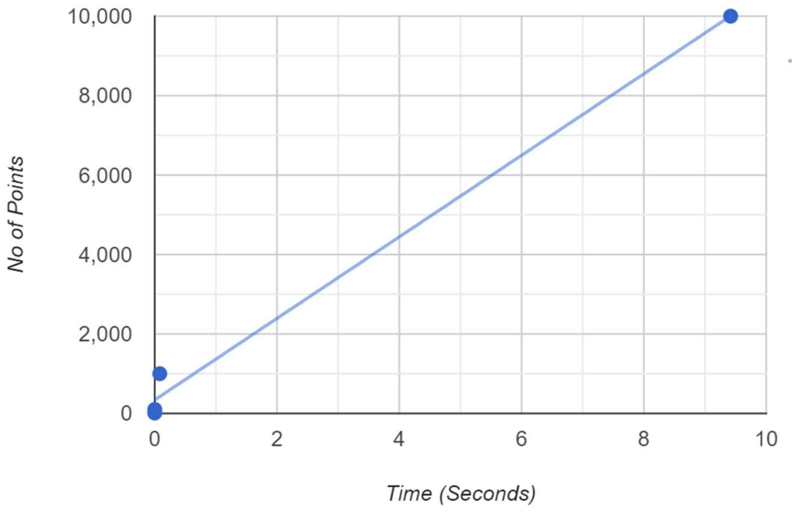
It is worth noting that << operator is not overloaded without friend keyword. The << operator is typically overloaded as a free function rather than a member function of the Point class.

Problem 3: Performance optimization. (10 marks)

Q1. Measure the execution time of your code for each value of n.

No of Points (n)	Execution Time (sec)
10	0
100	0.001
1000	0.087
10000	9.415

Q2. Draw a graph by hand of the execution time (y-axis) versus array size (x-axis). How doesyour code scale as n increases?



As the number of points (n) increases, the execution time of my code also increases as illustrated above. This suggests that the performance of my code doesn't scale well with larger datasets. As there is more data to process, the time taken by program to run becomes noticeably longer.

Problem 4: k-Means Clustering

The basic idea of what I can understand of k means algorithm is as follows:

1. Initialise the clusters - To do this, we randomly select k points which become markers, then assign each datapoint to its nearest marker point. The result of this is k clusters.
2. Compute the centroid of each cluster.
3. Assign each point to the nearest centroid and redefine the cluster.
4. Repeat steps 2 and 3.

In the k-means clustering algorithm, we repeatedly update the positions of cluster centres (centroids) and assign points to the nearest centroid. The idea is that there are only a limited number of ways we can group points into clusters, and with each update, we're guaranteed to make the overall clustering better. So, over time, the algorithm naturally converges to a stable and optimal solution.

I looked up to the code online and tried implementing it.

LAB VI

Problem 1

Q3. Let’s say I have the following code, will it use the type conversion operator or the overloaded subscript operator?

```
double arr_test[] = {1, 2};

Point p_test(arr_test);

cout << p_test[0] << "," << p_test[1]; // subscript operator is called

double *p_test_arr = p_test;

cout << p_test_arr[0] << "," << p_test_arr[1]; // conversion operator called
```

Problem 2

Q3. What is the output of the code? Explain why you get the output that you do.

```
Vector v1;

Point p1(1,1);

v1 = p1;

cout << v1 ;
```

We get the vector printed because we have already overloaded the ostream operatator.

LAB VII

Implemented all practical questions in the lab.

Problem 3

Q3. Speculate, if you can you call the print function from an object of type Location. Why orwhy not? Try it and see. What is the result?

If I try to call the print() function from an object of type Location, it will call the overridden print() function in the Location class. This is because the print() function is marked as virtual in the base class Element, and it is overridden in the derived class Location. The virtual keyword ensures that the overridden function in the most derived class is called.

LAB IX

Q1. Would you say that Java is a "successful" technology? What makes technology successful?

Ans. I think JAVA's use cases extend beyond my studies and because it is being taught to us and in the near future, we might be implementing it for solving real world problems as our initial goal to study this subject is, it is a successful technology. However, it might be possible as I pursue computer science further, I might not use it anytime in future. Beyond my subjective interpretation of Java being a successful technology depends on several factors. Java's byte code is portable (not in cpp) and it has active developer community contributing to a vast ecosystem of libraries, frameworks, and tools. It is used in web development, android development and cloud computing. So it is a successful technology.

Q2. List the full forms of the acronyms "JDK", "Java SE", "JRE". What do they mean?

And JDK stands for Java Developer Kit and is a software development kit used by developers to build, compile, and run Java applications.

Java Standard Edition includes the standard libraries, APIs, and the Java Virtual Machine (JVM) needed to run Java applications.

Java Runtime Environment is responsible for executing Java byte code.

Q3. What should you name the file?

Ans. The name of the file and public class should match and the first letter should be uppercase.

Rest all code implemented in lab twice by writing all code in a single file and by writing different class i.e Point and Element in different file. Noted that classes having default scope are accessible in main in spite of being in different files. Enjoyed learning java at first glance.

LAB X

Problem 1

Implemented Point_N and Element class in Java successfully.

Problem 2

Implemented the single threaded merge sort algorithm using the norm of the n-dimensional points. Here I used n=2 for simplicity purposes but it can be any value, the functionality has been provided for the same. Made random arrays of sizes 100, 1000, 10000, 100000 and measure the execution time of sort algorithm which is as follows-

Single-threaded merge sort:

Single-threaded Sorting time for array of size 10 is 0.2097 ms.

Single-threaded Sorting time for array of size 100 is 2.7704 ms.

Single-threaded Sorting time for array of size 1000 is 17.4206 ms.

Single-threaded Sorting time for array of size 10000 is 203.4139 ms.

Single-threaded Sorting time for array of size 100000 is 2183.662 ms.

Single-threaded Sorting time for array of size 1000000 is 18629.113100000002 ms.

Correctness of the sort using a dummy test case of size 10 was also ensured the points are being sorted based on their norm.

Problem 3

Now I studied the basics of multi-threaded and tried implementing it.

Dummy test case is working fine, points are sorted based on norms.

After multiple attempts of trying to find the execution time, I could sum up the following table. Point to note is that as the no of points increases, more threads help. A lot of factors of multiple threads and context switching overhead and also overhead of creation, deletion, managing, destroying threads are at work here.

Points	Time (ms)
10	0
100	0
1000	6
10000	44
100000	474
1000000	6226

LAB XI

Performance Gains Through Multithreading for Merge-Sort

Algorithm	Size(n)	Execution time (s)	Speedup
Single-threaded (baseline)	10	0	-
Multi-threaded	10	0	0
Multi-threaded w. manual optimization	10	1	0
Single-threaded (baseline)	100	2	-
Multi-threaded	100	0	Inconsistent
Multi-threaded w. manual optimization	100	0	Inconsistent
Single-threaded (baseline)	1000	17	-
Multi-threaded	1000	6	2.83
Multi-threaded w. manual optimization	1000	6	2.83

Single-threaded (baseline)	10,000	203	-
Multi-threaded	10,000	44	4.61
Multi-threaded w. manual optimization	10,000	41	4.95
Single-threaded (baseline)	100,000	2183	-
Multi-threaded	100,000	474	4.60
Multi-threaded w. manual optimization	100,000	459	4.75
Single-threaded (baseline)	1,000,000	18629	-
Multi-threaded	1,000,000	6226	2.99
Multi-threaded w. manual optimization	1,000,000	5959	3.12

Manual optimisations that I did was as follows –

1. Computing the length of arrays outside the loop.
2. Computing the norm only once for each element.

As can be observed from the table it leads to reduction in execution time. When we compare normal multithread and multi thread with manual optimisations, there is decrement in execution time as well.

Q2. Multi thread merge sort of what I understood.

Ans. The program utilizes two threads to concurrently sort two halves of the array using separate threads and then merges the results. The use of threads allows for parallelization of the sorting process, potentially improving overall performance by decreasing execution time. Runnable interface has been used to facilitate the creation and running of threads. class MultiThreadedMerge contains declares a member variable t of type Thread associated with current object which is basically subarray. he array is initially split into two halves (subArr1 and subArr2).Two threads, t1 and t2, are created using the MultiThreadedMerge class. Each thread is responsible for sorting one of the subarrays. Each thread (t1 and t2) is assigned a portion of the array (subArr1 and subArr2, respectively) to sort. The thread initialization involves setting up the subarray and creating a new thread object.

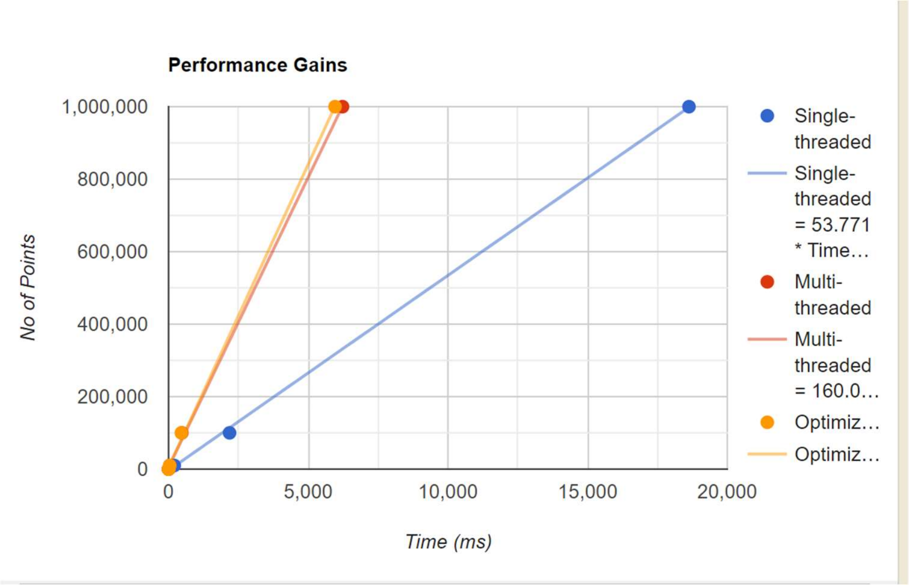
Basically two threads-t1 and t2 has been spawned, each representing a separate thread for sorting a subarray.

The work is divided by splitting the input array into two halves (subArr1 and subArr2). Each thread (t1 and t2) is assigned to sort one of these subarrays concurrently.

The Java scheduler manages the execution of threads. When we start t1.t and t2.t, the scheduler decides when each thread will run. The join() method is used to ensure that the main thread waits for t1 and t2 to complete their sorting before proceeding to merge the results.

Q3. Since multi-threaded merge sort is a simple small program with only two threads, no interrupts occur. The number of interruptions will depend on the specific execution of the program and the system it is running on.

Q4.


Performance Gains
1,000,000
800,000
600,000
400,000
200,000
0
No of Points
0
5,000
10,000
15,000
20,000
Time (ms)
● Single-threaded
Single-threaded = 53.771 * Time…
● Multi-threaded
Multi-threaded = 160.0…
● Optimiz…
Optimiz…