DSBA Introduction to Programming
Seminar #17 Wed 09.03.2022
Exercise

Create a class Vector3D that represents three-dimensional point (x,y,z) in the Cartesian plane,
and complete the class methods, as well as operator overloads, as indicated by each of the tasks below.

```
class Vector3D
{
        public:
                /* class methods */
        private:
                /* attributes */
                double _x;
                double _y;
                double _z;
};
```

Task 1 – Empty Constructor

Create the empty constructor Vector3D() where every attribute is set to 0.0 (_x = 0.0, _y = 0.0 and _z = 0.0),

Task 2 – Constructor with Arguments

Code the constructor Vector3D(double x, double y, double z) that sets every attribute of the vector according to the arguments (_x = x, _y = y, _z = z).

Task 3 – Copy Constructor

Code the constructor Vector3D(const Vector3D& v2) that sets every attribute of the vector according to the attributes of an input vector v2 (for example, _x = v2._x)

Task 4 – Code getters/setters

For every attribute of the vector, code a method that returns the attribute (a *getter*) and a method that sets the value of the attribute according to a given argument (a *setter*)

For example, for attribute _x you should do:

```
// getter
double getX() const
{
        return x;
}
```

```
// setter
void setX(double x)
{
        _x = x;
}
```

Then, you are to code a similar *getter* and a similar *setter* for attributes _y and _z

Task 5 – Overload the + operator

Code the overload of the operator + as follows: Vector3D operator+ (const Vector3D& v1, Vector3D& v2) where you should return a new vector v3 whose attributes will be the sum of v1 and v2's attributes.

For example for v3._x, you implement as follows: v3.setX(v1.getX() + v2.getX());

Task 6 – Overload the * operator

Code the overload of the operator * as follows: double operator* (const Vector3D& v1, Vector3D& v2) where you should return the dot product between vectors v1 and v2.

Task 7 – Overload the * operator (yes, once more, but with another input argument!) - scalar product

Code the overload of the operator * as follows: `Vector3D operator* (const Vector3D& v1, double d)`
where you should return a new vector v2 whose attributes will be multiplication of every attribute of v2 by d

For example, for v2._x you implement as follows:

`v2.setX( v1.getX() * d );`

Task 8 – Vector magnitude

Code inside the class a function double magnitude() to get the magnitude of the Vector3D defined as:

$$\sqrt{(x^2+y^2+z^2)}$$

Task 9 – Overload the < operator

Overload the operator< as follows: `bool operator<(const Vector3D& v1, const Vector3D& v2)` where:

vector v1 < vector v2 iff `v1.magnitude() < v2.magnitude()`

Task 10 – Overload << operator

Overload the print operation << for a Vector3D such that to print a vector in the terminal in format (x,y, z)

For example, if we have that _x = 3.4, _y = 1.0, _z = -1.4 then you print the terminal

(3.4,1.0,-1.4)

Task 11 – Overload >> operator

Overload the read operation >> for a Vector3D such that to read a vector from the terminal.

For example, if the user puts in the terminal 3.4 1.0 -1.4

then you create an object Vector3D with attributes _x = 3.4, _y = 1.0, _z = -1.4

Final Task

Part 1

Create a cointaner `std::multiset<Vector3D> s` where you will insert 100 Vector3D objects,

such that for every Vector3D, its attributes _x _y _z are random numbers generated in the interval [-1.0, 1.0]

For generating random real numbers, you may use the class std::uniform_real_distribution from C++
https://www.cplusplus.com/reference/random/uniform_real_distribution/

Part 2

Print all elements stored in the container s.

Part 3

Calculate and print the average of the magnitude of all vectors contained in s.