# COMP3512 - Lab Exercise 4 (Oct 2 - 6, 2017)

This is an exercise that you need to do on a computer. You'll need to commit and push your code to your GitLab repo, and submit for automated marking via Slack.

For this exercise, you will need to write a C++ class Vector that will be used to perform various vector operations.

## 1. Project Setup

1. Open Lab4.sln in Visual Studio 2017
2. Add Vector.h file to your project. (refer to Lab 1 if you don't know how)
3. Add the following content in the header file.

#pragma once

namespace lab4

{

class Vector

{

public:

Vector(float x, float y, float z);

virtual ~Vector();

Vector operator+(const Vector& other) const;

Vector operator-(const Vector& other) const;

float operator\*(const Vector& other) const;

Vector operator\*(float operand) const;

float operator[](unsigned int index) const;

float GetX() const;

float GetY() const;

float GetZ() const;

friend Vector operator\*(float operand, const Vector& v);

private:

// private variables here

};

}

1. Add Vector.cpp file to your project.
2. Add the following empty implementations in the cpp file.

#include "Vector.h"

namespace lab4

{

Vector::Vector(float x, float y, float z)

{

}

Vector::~Vector()

{

}

Vector Vector::operator+(const Vector& other) const

{

// implement vector addition here

}

Vector Vector::operator-(const Vector& other) const

{

// implement vector subtraction here

}

float Vector::operator\*(const Vector& other) const

{

return 0.f; // implement dot product here

}

Vector Vector::operator\*(float operand) const

{

// implement scalar multiplication of vector

}

float Vector::operator[](unsigned int index) const

{

return 0.f; // implement subscription operator here

}

float Vector::GetX() const

{

return 0.f; // return x component

}

float Vector::GetY() const

{

return 0.f; // return y component

}

float Vector::GetZ() const

{

return 0.f; // return z component

}

Vector operator\*(float operand, const Vector& v)  
{  
 // implement scalar multiplication of vector  
}

}

1. Note that the solution will not build because some methods are not returning the right types. You will have to implement them to be able to build the solution.

### Expected Behavior of Vector class

* ‘+’ operator should perform memberwise addition. ex> [1, 2, 3] + [2, 4, 1] = [3, 6, 4]
* ‘-’ operator should perform memberwise subtraction. ex> [1, 2, 3] - [2, 4, 1] = [-1, -2, 2]
* ‘\*’ operator where argument is Vector type should calculate a dot product. (<https://en.wikipedia.org/wiki/Dot_product>).

ex>

lab4::Vector a(1, 2, 3);

lab4::Vector b(2, 3, 1);

A \* b // Should give (1 \* 2) + (2 \* 3) + (3 \* 1) = 11

* ‘\*’ operator where argument is a float should perform scalar multiplication. ex> [1, 2, 3] \* 4 = [4, 8, 12]
* ‘[]’ operator should return X, Y and Z component of a vector for indexes 0, 1 and 2 consecutively. If the index is greater than 2, then it should always return Z component.

ex>

lab4::Vector a(1, 2, 3);

a[0] // 1

a[1] // 2

a[2] // 3

a[6] // 3

* GetX(), GetY() and GetZ() should return the Vector’s respective components.
* Global ‘\*’ operator (in Vector.h file at the bottom. Don’t miss it!) performs scalar multiplication where left side of the operand is the scalar value. ex> 4 \* [1, 2, 3] = [4, 8, 12]

## 2. Implement All Class Functions Introduced in Step 1

* You can test your code using something like the following code in main.cpp

#include "Vector.h"

#include <iostream>

int main()

{

lab4::Vector a(4, -5, 6);

lab4::Vector b(7, 2, -6);

lab4::Vector sum = a + b;

lab4::Vector difference = a - b;

float dotProduct = a \* b;

lab4::Vector scalarProduct = a \* 2;

lab4::Vector scalarProduct2 = 2 \* a;

std::cout << difference[2] << std::endl;

return 0;

}

## 3. Commit, Push and Ask for a Build

You know the drill :)

# 