<u>QUESTIONS</u>

Write a C++ program to implement the V3D Class.

i) Define **three double private** data member for x, y, and z coordinates of the vector such as x, y, and z and **a private const string** data member name. Also, define **a private static int** count data member. Initialize the static data member with zero.
ii) The constructor of the class is default constructor with values 0.0 for x, y, and z. **In constructor, use set functions.** Lastly, print an information message about the object.
iii) The destructor of the class must include an information message about the object.
iv) Include *setX* **public** member function which receives x coordinate and returns nothing.
v) Include *setY* **public** member function which receives y coordinate and returns nothing.
vi) Include *setZ* **public** member function which receives z coordinate and returns nothing.
vii) Include *getX* **public** member function which receives nothing and returns x coordinate.
viii) Include *getY* **public** member function which receives nothing and returns y coordinate.
ix) Include *getZ* **public** member function which receives nothing and returns z coordinate.
x) Include *getName* **public** member function which receives nothing and returns the name of the vector.
xi) Include *getCount* **public** member function which receives nothing and returns count data member.
xii) Include *print* **public** function member function, which receives nothing and returns nothing. In print function, **use get functions** and print the vector following format:

name(x,y,z)

xiii) Include *algebra* **public** member function, which receives a V3D by reference and returns a V3D by reference (i.e V3D &algebra (const V3D &b)). In the function, assume that current **(this)** object is represented with $\vec{a}$ and given V3D by reference ($\vec{b}$). Then, calculate $\vec{a} - 4\vec{b}$ and return current **(this)** object by reference.
xiv) Include *crossProduct* **public** member function, which receives a V3D by reference and returns a V3D (i.e V3D crossProduct (const V3D &b)). In the function, assume that current **(this)** object is represented with $\vec{a}$, the given V3D by reference ($\vec{b}$). Then, calculate $\vec{a}x\vec{b}$ and **it is assigned to a new object**. Then, **return the new object** (**Hint**: The current **(this)** object must not change).

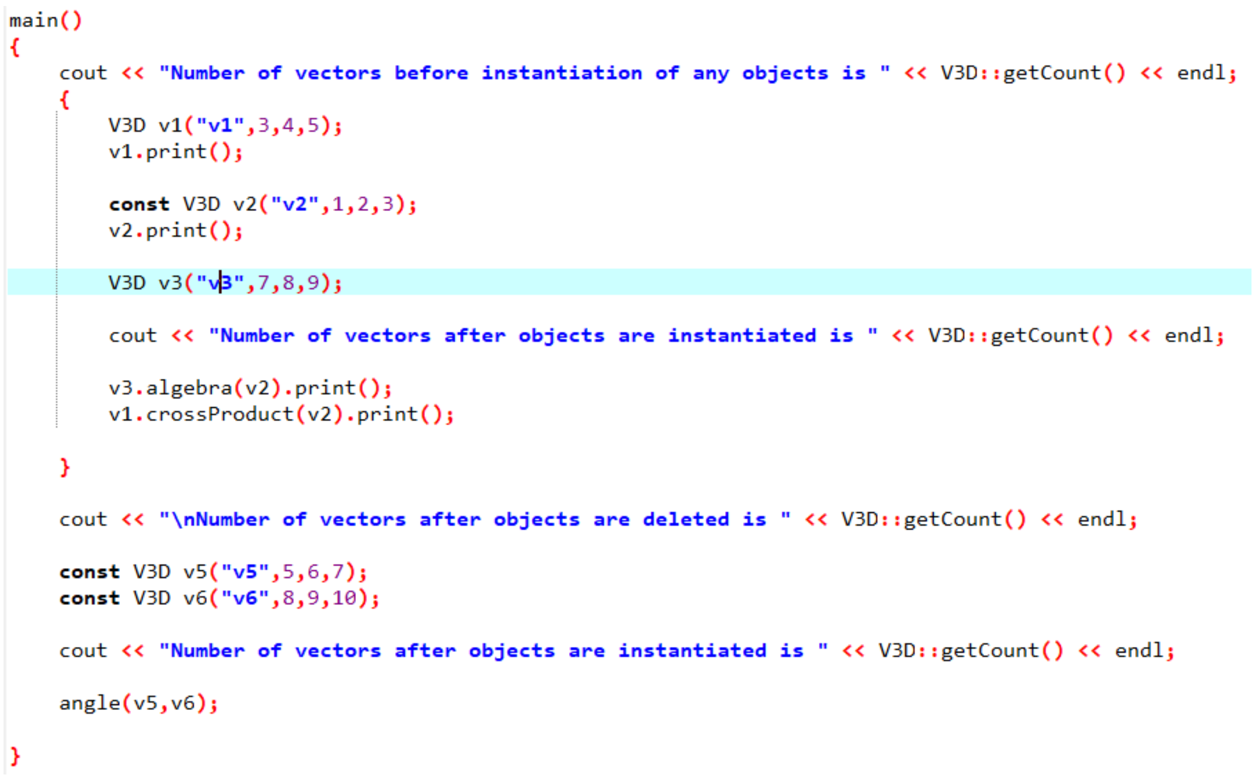
$$\vec{a}x\vec{b} = (a_y b_z - a_z b_y, a_z b_x - a_x b_z, a_x b_y - a_y b_z)$$

xv) Include *angle* **friend** function which receives two V3D and returns nothing (i.e **friend void** *angle* (const V3D &v1, const V3D &v2)). In function, **calculate** and **print** the angle between of two vectors.Also **print** the v1 and v2 objects. Use the following formula:

$$cos\theta = \frac{\overrightarrow{v1}.\overrightarrow{v2}}{||v1||||v2||}$$

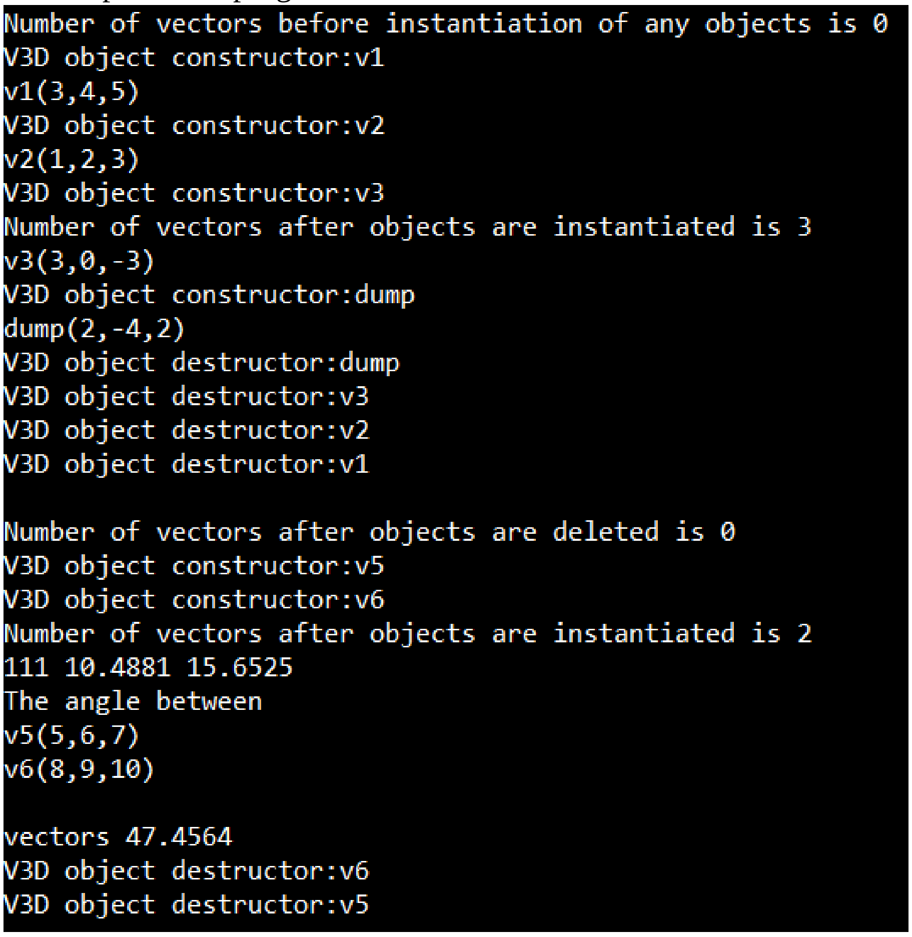
Draw UML Class Diagram for V3D class. Remember that, friend functions are not member functions of any class.

Test your program with the following driver program.



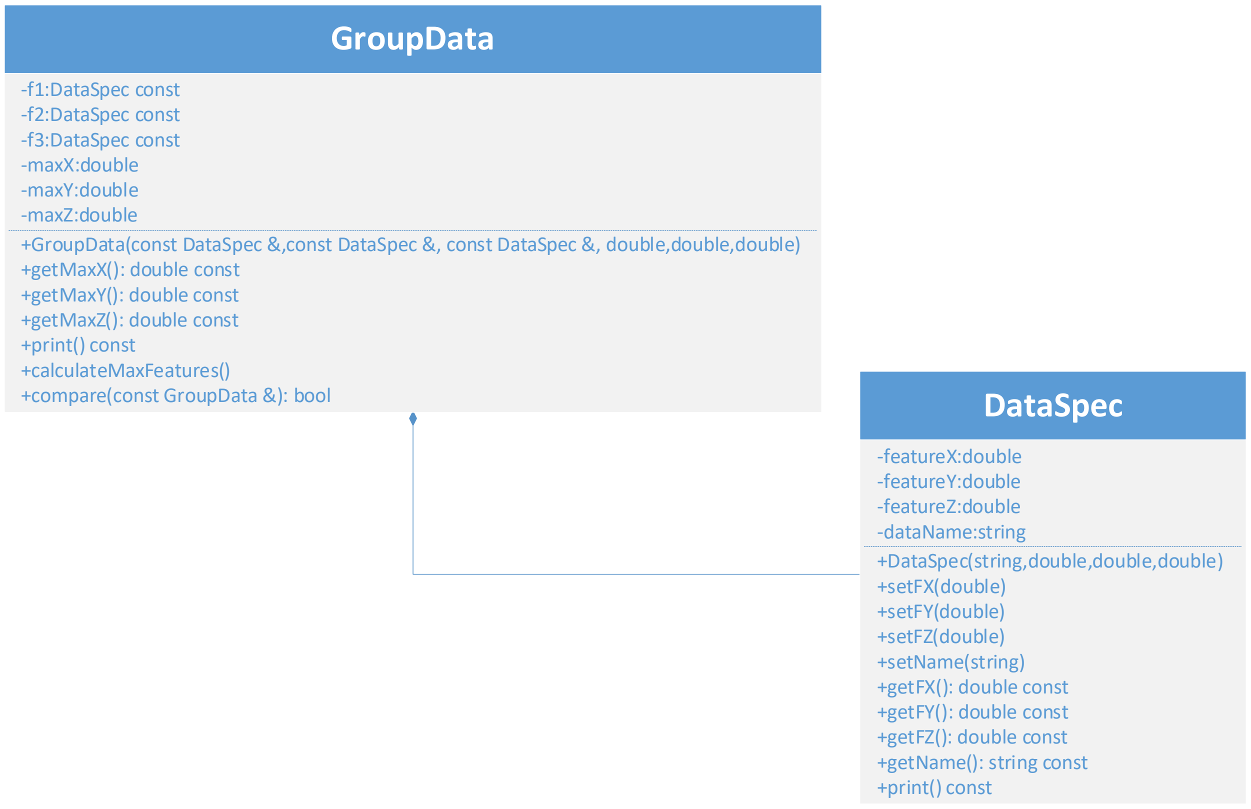
```
main()
{
    cout << "Number of vectors before instantiation of any objects is " << V3D::getCount() << endl;
    {
        V3D v1("v1",3,4,5);
        v1.print();

        const V3D v2("v2",1,2,3);
        v2.print();

        V3D v3("v3",7,8,9);

        cout << "Number of vectors after objects are instantiated is " << V3D::getCount() << endl;

        v3.algebra(v2).print();
        v1.crossProduct(v2).print();

    }

    cout << "\nNumber of vectors after objects are deleted is " << V3D::getCount() << endl;

    const V3D v5("v5",5,6,7);
    const V3D v6("v6",8,9,10);

    cout << "Number of vectors after objects are instantiated is " << V3D::getCount() << endl;

    angle(v5,v6);

}
```

The output of the program must be as follows:



```
Number of vectors before instantiation of any objects is 0
V3D object constructor:v1
v1(3,4,5)
V3D object constructor:v2
v2(1,2,3)
V3D object constructor:v3
Number of vectors after objects are instantiated is 3
v3(3,0,-3)
V3D object constructor:dump
dump(2,-4,2)
V3D object destructor:dump
V3D object destructor:v3
V3D object destructor:v2
V3D object destructor:v1

Number of vectors after objects are deleted is 0
V3D object constructor:v5
V3D object constructor:v6
Number of vectors after objects are instantiated is 2
111 10.4881 15.6525
The angle between
v5(5,6,7)
v6(8,9,10)

vectors 47.4564
V3D object destructor:v6
V3D object destructor:v5
```

The UML Class Diagram for GroupData Class is given in the following figure. GroupData Class composes DataSpec Class. Write a C++ program to implement the GroupData Class.



DataSpec Class

i) The constructor of the class is a default constructor with values 0.0 for featureX, featureY, and featureZ. **In constructor, use set functions and print the object name.**

ii) In print function, **use get functions** to print the point in the following format **name(featureX,featureY,featureZ)**.

GroupData Class

i) The constructor of the class is a default constructor with values 0.0 for maxX, maxY, and maxZ.

ii) In print function, **use get functions** to print the maxX, maxY, and maxZ data members.

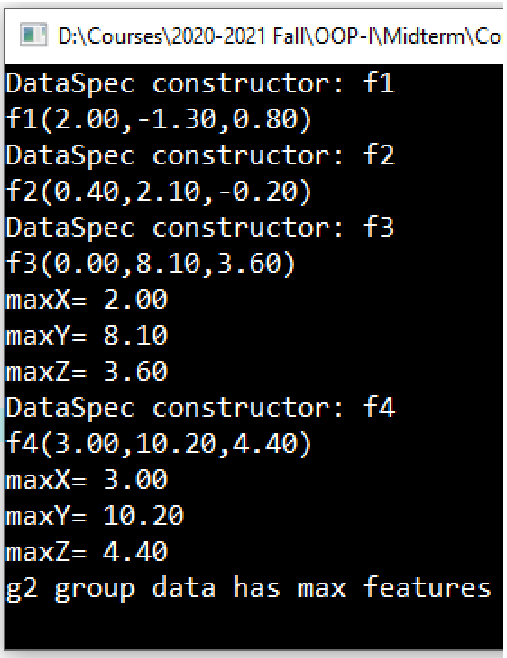
iii) Include *calculateMaxFatures* function, calculate the maxX, maxY, and maxZ data members. Assume that we have three features (i.e f1, f2, and f3) and **use get functions**. Use the following formulas:

$$\text{maxX} = \max(f1.x, f2.x, f3.x)$$
$$\text{maxY} = \max(f1.y, f2.y, f3.y)$$
$$\text{maxZ} = \max(f1.z, f2.z, f3.z)$$

iv) Include *compare* function, compare DataGroup objects in terms of maxX, maxY, and maxZ data members. If, maxX, maxY, and maxZ data members of g1 is greater than maxX, maxY, and maxZ data members of g2, return true. Otherwise, return false.

Test your program with the following driver program.

```
main() {
    const DataSpec f1("f1",2.0,-1.3,0.8);
    f1.print();
    const DataSpec f2("f2",0.4,2.1,-0.2);
    f2.print();
    const DataSpec f3("f3",0.0,8.1,3.6);
    f3.print();

    GroupData g1(f1, f2 ,f3);
    g1.calculateMaxFeatures();
    g1.print();

    const DataSpec f4("f4",3.0,10.2,4.4);
    f4.print();

    GroupData g2(f1, f2 ,f4);
    g2.calculateMaxFeatures();
    g2.print();

    if (g1.compare(g2))
        cout << "g1 group data has max features" << endl;
    else
        cout << "g2 group data has max features" << endl;
}
```


D:\Courses\2020-2021 Fall\OOP-I\Midterm\Co

```
DataSpec constructor: f1
f1(2.00,-1.30,0.80)
DataSpec constructor: f2
f2(0.40,2.10,-0.20)
DataSpec constructor: f3
f3(0.00,8.10,3.60)
maxX= 2.00
maxY= 8.10
maxZ= 3.60
DataSpec constructor: f4
f4(3.00,10.20,4.40)
maxX= 3.00
maxY= 10.20
maxZ= 4.40
g2 group data has max features
```