Write a class Matrix with the following capabilities:

i) Define an integer 2D vector data member (i.e a). **Do not give dimension in header.**

ii) In constructor resize the vector for 2-by-2 vector. Also initialize the vector with zero. **Do not use two nested for loop for initialization. Otherwise, you will get 0 credits.**

iii) Include overloaded binary addition and multiplication operators for this class. Addition operator will perform matrix summation. Multiplication operator will perform scalar multiplication. **Overloaded operators must be public member functions**. **Use two nested for loop in functions. Otherwise, you will get 0 credits.**

iv) Include overloaded unary minus and negation operators (-, !) for this class. Minus operator will calculate inverse of a matrix. Negation operator will calculate transpose of a matrix. **Overloaded operators must be non-member (friend) functions**.

v) Include overloaded the stream extraction operator function operator (>>) for this class. **Use two nested for loop in function. Otherwise, you will get 0 credits.**

vi) Include overloaded the stream insertion operator function operator (<<) for this class. **The format for output is given below. Use two nested for loop in function. Otherwise, you will get 0 credits.**

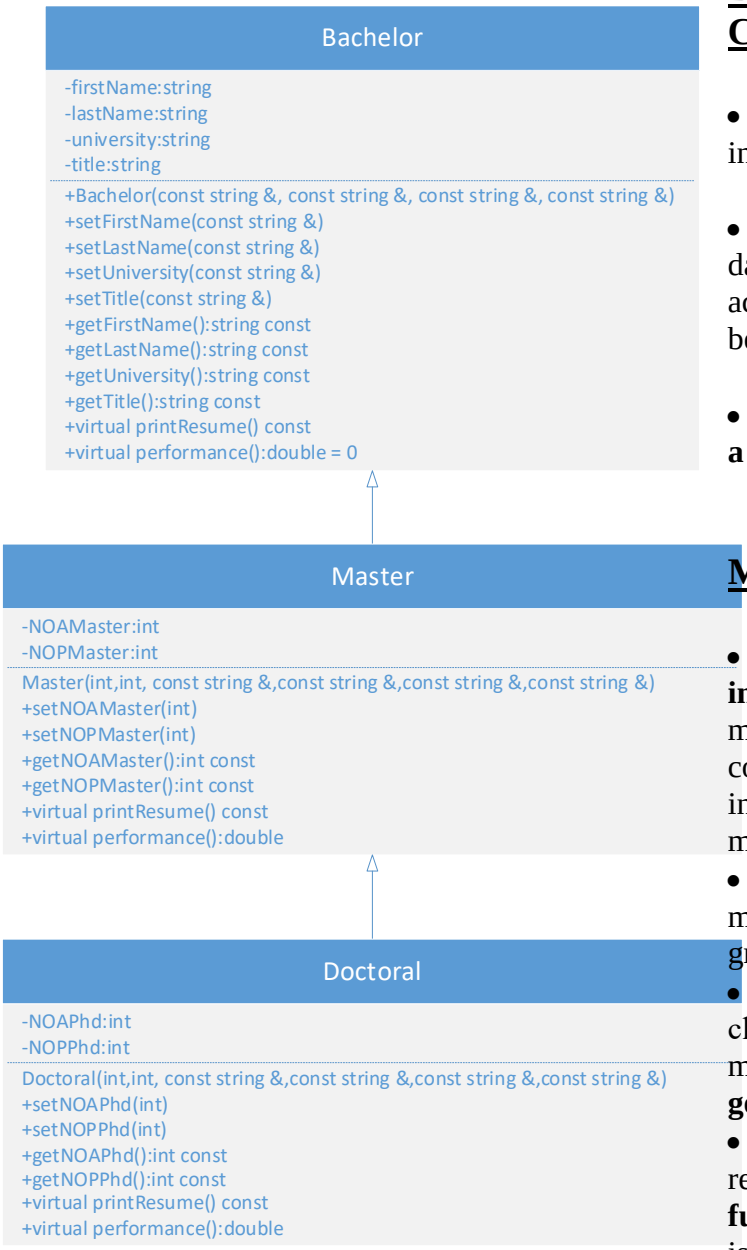
Test your program with the following driver program.

```
main()
{
    Matrix m1,m2,x;

    cout << "Enter matrix items" << endl;
    cin >> m1 >> m2;

    cout << "\nm1\n" << m1 << endl;
    cout << "\nm2\n" << m2 << endl;
    cout << "\nx\n" << x << endl;

    x=m1+m2;
    cout << "\nm1+m2\n" << x << endl;

    x=m1*5;
    cout << "\nx\n" << x << endl;

    -m2;
    cout << "\nInverse of m2\n" << m2 << endl;

    !m1;
    cout << "\nTranspose of m1\n" << m1 << endl;
}
```

The output of the program must be as follows:

```
Enter matrix items
1 2 3 4 3 8 11 4

m1
        |  1   2 |
        |  3   4 |


m2
        |  3   8 |
        | 11   4 |


x
        |  0   0 |
        |  0   0 |


m1+m2
        |  4  10 |
        | 14   8 |


x
        |  5  10 |
        | 15  20 |


Inverse of m2
        |  0   0 |
        |  0   3 |


Transpose of m1
        |  1   3 |
        |  2   4 |
```

Write a C++ code to implement the UML class diagram.



Bachelor Class (Abstract Base Class)

- In constructor, use **set functions** to initialize data members.
- In printResume function, print the data members by using **get functions** according to the output, which is given below.
- **Notice that performance function is a pure virtual function.**

Master Class

- In constructor, use **base-class initializer syntax** to initialize data members of the base class. In the constructor, use **set functions** to set initial values of the Master class data members.
- In set functions, validate the data members of the class. They must be greater than 0.
- In printResume function, call the base class's print function. Then print data members of the Master Class by using **get functions**.
- In performance function, calculate and return performance value by using **get functions**. The formula for performance is given below:

$$performMaster = 12.0 + 6.2 * NOAMaster + 14.5 * NOPMaster$$

Doctoral Class

• In constructor, use **base-class initializer syntax** to initialize data members of the base class. In the constructor, use **set function** to set initial values of the Doctoral class data member.

• In set functions, validate the data. Data members of the class must be greater than 0.

• In printResume function, call the Master class's printResume function. Then print data member of the Doctoral Class by using **get functions**.
• In performance function, calculate and return performance. In the function, call Master class's performance function and **get functions**. The formula for points is given below:

$$performPhd = performMaster + 8.5 * NOAPhd + 26.5 * NOPPhd$$

Driver Program

• Define two objects for Master class and initialize the objects with data given below.
• Define two objects for Doctoral class and initialize the objects with data given below.
• Define a **Bachelor *** vector with 4 items. Initialize the vector with the four objects defined for Master and Doctoral classes.
• **Use a for loop to call print and performance methods**.

The output of the program must be as follows:

```
Name: Ricardo
Surname: Charles
Title: Research Asistant
NOA Master: 16
NOP Master: 3
Performance 154.7

Name: Jonas
Surname: Benz
Title: Research Asistant
NOA Master: 4
NOP Master: 1
Performance 51.3

Name: Andrew
Surname: Liang
Title: Assistant Professor
NOA Master: 26
NOP Master: 8
NOA Phd: 18
NOP Phd: 13
Performance 786.7

Name: Lilian
Surname: Bartez
Title: Professor
NOA Master: 13
NOP Master: 2
NOA Phd: 23
NOP Phd: 5
Performance 449.6
```

Assume that we have point cloud data that include x, y, and z coordinates. Write a C++ program to implement following steps.

i) Define a structure (i.e., **struct Point**) to store each point cloud data. The structure includes x, y, and z coordinates.

ii) Define a **Point vector** with 20 items.

iii) Use a generator function (i.e., **Point initPoint (void)**) and appropriate algorithm to initialize point vector. In the function, initialize x, y, and z coordinates randomly in interval [0-10] with step size 0.1 (**Hint: Do not use loop)**.

iv) The program must include printPoint function that receives point vector. In the function print vector items. An example output given in the figure.

v) Define an **integer vector** with 20 items for **mask**. In the vector, the items that are z member greater than 5 will be 1. Other items will be 0.

vi) Use a predicate function (i.e., **bool zGT5 (Point)**) and appropriate algorithm to obtain mask vector (**Hint: Do not use loop)**.

vii) Print the mask vector.

viii) Define a map. The keys of points subscript of mask vector. You must use the following typedef to define the map. Assign point data corresponding to 1's in mask vector.

typedef map<int, Point > MiP;

ix) Print the map.

The output of the program must be as follows:

```
Point Vector
  X    Y    Z
4.1  6.7  3.4
0.0  6.9  2.4
7.8  5.8  6.2
6.4  0.5  4.5
8.1  2.7  6.1
9.1  9.5  4.2
2.7  3.6  9.1
0.4  0.2  5.3
9.2  8.2  2.1
1.6  1.8  9.5
4.7  2.6  7.1
3.8  6.9  1.2
6.7  9.9  3.5
9.4  0.3  1.1
2.2  3.3  7.3
6.4  4.1  1.1
5.3  6.8  4.7
4.4  6.2  5.7
3.7  5.9  2.3
4.1  2.9  7.8

Mask Vector
0 0 1 0 1 0 1 1 0 1 1 0 0 0 1 0 0 1 0 1

Point Map
Key    X     Y     Z
  2  7.8   5.8   6.2
  4  8.1   2.7   6.1
  6  2.7   3.6   9.1
  7  0.4   0.2   5.3
  9  1.6   1.8   9.5
 10  4.7   2.6   7.1
 14  2.2   3.3   7.3
 17  4.4   6.2   5.7
 19  4.1   2.9   7.8
```