**ASSIGNMENT # 04**

**MOHAMMAD BASIL ALI KHAN**

**20K-0477**

Question#01:

(a)

#include<iostream>

using namespace std;

class A{

    private:

    int x;

    public:

    A(int x)

    {

        this->x=x;

    }

    void display()

    {

        cout << "Incremented value: " << x;

    }

    void operator ++ (int )

    {

        x++;

    }

};

int main()

{

    A Obj(5);

    Obj++;

    Obj.display();

    return 0;

}

(b)

#include<iostream>

using namespace std;

class Assignment{

    private:

    int x;

    public:

    void setData(int a)

    {

        x = a;

    }

    Assignment operator + (int val)

    {

        Assignment temp;

        temp.x = x + val;

        return temp;

    }

    void Display()

    {

        cout << "Obj + 20 : " << x << endl;

    }

};

int main()

{

    Assignment Obj, Obj1;

    Obj.setData(10);

    Obj1 = Obj + 20;

    Obj1.Display();

    return 0;

}

( c )

#include <iostream>

using namespace std;

class Assignment{

    private:

    int x;

    public:

        void setData(int a)

        {

            x = a;

        }

        Assignment operator \*(Assignment Obj)

        {

            Assignment temp;

            temp.x = x \* Obj.x;

            return (temp);

        }

        void Display()

        {

            cout<<"Obj3 = Obj1 \* Obj2 : "<< x << endl;

        }

};

int main()

{

    Assignment Obj1, Obj2, Obj3;

    Obj1.setData(5);

    Obj2.setData(6);

    Obj3 = Obj1 \* Obj2;

    Obj3.Display();

}

(d)

#include<iostream>

using namespace std;

class Assignment{

    private:

    int x;

    public:

    void setData(int a)

    {

        x = a;

    }

    bool operator <(Assignment Obj)

    {

        if(x < Obj.x)

        {

            return true;

        }

        else

        {

            return false;

        }

    }

};

int main()

{

    Assignment Obj1, Obj2;

    Obj1.setData(5);

    Obj2.setData(12);

    if(Obj1 < Obj2)

    {

        cout << "Obj1 is less than Obj2.." << endl;

    }

    else

    {

        cout << "Obj2 is less than Obj1.." << endl;

    }

}

( e )

#include<iostream>

using namespace std;

class Assignment{

    private:

    int x;

    public:

    void setData(int a)

    {

        x = a;

    }

    bool operator ==(Assignment Obj)

    {

        if(x == Obj.x)

        {

            return true;

        }

        else

        {

            return false;

        }

    }

};

int main()

{

    Assignment Obj1, Obj2;

    Obj1.setData(5);

    Obj2.setData(5);

    if(Obj1 == Obj2)

    {

        cout << "Obj1 is equal Obj2.." << endl;

    }

    else

    {

        cout << "Obj1 is not equal to Obj2.." << endl;

    }

}

(f)

#include<iostream>

using namespace std;

class Assignment{

    private:

    int x;

    public:

    void setData(int a)

    {

        x = a;

    }

    Assignment operator \* (Assignment Obj)

    {

        Assignment temp;

        temp.x = x \* Obj.x;

        return temp;

    }

    Assignment operator / (Assignment Obj)

    {

        Assignment temp;

        temp.x = x / Obj.x;

        return temp;

    }

    bool operator > (Assignment Obj)

    {

        if(x > Obj.x)

        {

            return true;

        }

        else

        {

            return false;

        }

    }

};

int main()

{

    Assignment Obj1, Obj2, Obj3;

    Obj1.setData(3);

    Obj2.setData(4);

    Obj3.setData(2);

    if( (Obj1 \* Obj2) > (Obj2 / Obj3) )

    {

        cout << "(Obj1 \* Obj2) > (Ob2 / Obj3)" << endl;

        cout << "Proved !!! " << endl;

    }

    else

    {

        cout << "(Obj1 \* Obj2) > (Ob2 / Obj3)" << endl;

        cout << "Not Proved !!! " << endl;

    }

}

Question # 02:

* **Friend Function:**

Friend functions are needed when we have to access private or protected members of class.

**Coding Example:**

#include<iostream>

using namespace std;

class Addition{

    private:

    int num1;

    int num2;

    public:

    void setData(int x, int y)

    {

        num1 = x;

        num2 = y;

    }

    friend int ADD (Addition Obj);

};

int ADD(Addition Obj)

{

    int temp;

    temp = Obj.num1 + Obj.num2;

    return temp;

}

int main()

{

    Addition Obj1;

    Obj1.setData(3,4);

    cout << "Result after adding two class private members: " << ADD(Obj1) << endl;

}

* **Static Function:**

We need static function when we have to access static data members and other static member function. A static member function can be called even if no objects of the class exist and the static functions are accessed using only the class name and the scope resolution operator ::.

**Coding Example:**

#include<iostream>

using namespace std;

class Employee{

    private:

    string name;

    int salary;

    static int count;

    public:

    Employee(string name, int salary)

    {

        this->name = name;

        this->salary = salary;

        count++;

    }

    static int getcount()

    {

        return count;

    }

    void displayData()

    {

        cout << "Employee Name: " << name << endl;

        cout << "Salary: " << salary << endl;

    }

};

int Employee :: count = 0;

int main()

{

    Employee Obj1("Basil", 20000);

    cout << "Number Of Employee: " << Employee :: getcount() << endl;

    Obj1.displayData();

    Employee Obj2("Ahmed", 30000);

    Obj2.displayData();

    cout << "Number Of Employee: " << Employee :: getcount() << endl;

}

* **Virtual Function:**

When we have to access derived class object using pointer or reference to the base class, we need virtual function for that object to execute the derived class version of function. We use virtual function when run time polymorphism is done.

**Coding Example:**

#include<iostream>

using namespace std;

class Car{

    protected:

    string color;

    int model;

    int price;

    int NoOfDoors;

    public:

    virtual void display()

    {

        cout << "Base Class Function...." << endl;

    }

};

class Audi : public Car{

    public:

    Audi(string col, int mod, int p, int doors)

    {

        color = col;

        model = mod;

        price = p;

        NoOfDoors = doors;

    }

    void display()

    {

        cout << "Audi Car Details: " << endl;

        cout << "Color: " << color << endl;

        cout << "Model: " << model << endl;

        cout << "No of Doors: " << NoOfDoors << endl;

        cout << "Price: " << price << " lakhs" << endl;

    }

};

class Ferrari : public Car{

    public:

    Ferrari(string col, int mod, int p, int doors)

    {

        color = col;

        model = mod;

        price = p;

        NoOfDoors = doors;

    }

    void display()

    {

        cout << "Ferrari Car Details: " << endl;

        cout << "Color: " << color << endl;

        cout << "Model: " << model << endl;

        cout << "No of Doors: " << NoOfDoors << endl;

        cout << "Price: " << price << " lakhs" << endl;

    }

};

int main()

{

    Car \*ptr[2];

    Audi obj1("Black", 2021, 167, 4);

    Ferrari obj2("Red", 2021, 320, 4);

    ptr[0]=&obj1;

    ptr[1]=&obj2;

    ptr[0]->display();

    ptr[1]->display();

    return 0;

}