**ADVANCED ASSIGNMENT - 3**

Name: **HITARTH GANATRA**

Sem: **II**

Div.: **1**

ID: **22BCA064**

Subject: **Fundamentals of Object Oriented Programming Language**

Q1

**CODE:**

*// 1.   Write a program to swap two characters of different data types using the function overloading concept.*

#include<iostream>

using namespace std;

class Swap{

    int x, y;

    char a, b;

    public:

        void swap(int x, int y){

            cout << "Before : ";

            cout << "x = " << x << " y = " << y << endl;

            int temp = x;

            x = y;

            y = temp;

            cout << "After : ";

            cout << "x = " << x << " y = " << y << endl;

        }

        void swap(char a, char b){

            cout << "Before : ";

            cout << "a = " << a << " b = " << b << endl;

            char temp = a;

            a = b;

            b = temp;

            cout << "After : ";

            cout << "a = " << a << " b = " << b << endl;

        }

};

int main(){

    Swap func, func1;

    func.swap(10, 20);

    func1.swap('A', 'B');

    return 0;

}

Q2

**CODE:**

*// 2.   Write a program to calculate the volume of square(s\*s\*s), circle (4/3\* pi\*r\*r\*r), and rectangle(l\*w\*h) using the function overloading concept.*

#include<iostream>

using namespace std;

class Area{

    public:

        void area(float s){

            cout << "The Area of Square is : " << (s\*s\*s) << endl;

        }

        void area(float pi, float r){

            cout << "The Area of Circle is : " << (4/3 \* pi \*r\*r\*r) << endl;

        }

        void area(float l, float w, float h){

            cout << "The Area of rectangle is : " << (l\*w\*h) << endl;

        }

};

int main(){

    Area square, circle, rect;

    square.area(10);

    circle.area(3.14, 10);

    rect.area(3, 4, 5);

    return 0;

}

Q3

**CODE:**

*// 3.   Write a program to overload unary minus operator*

#include<iostream>

using namespace std;

class Demo{

    public:

        int a, b, c;

        Demo(int a, int b, int c){

*this*-> a = a;

*this*-> b = b;

*this*-> c = c;

        }

        int operator - (){

            a = -a;

            b = -b;

            c = -c;

        }

        void display();

};

void Demo :: display (){

    cout << "A : " << a << endl;

    cout << "B : " << b << endl;

    cout << "C : " << c << endl << endl;

}

int main()

{

    Demo d1(10, -20, 30);

    d1.display();

    -d1;

    d1.display();

    return 0;

}

Q4

**CODE:**

*// 4.   Write a program to overload pre decrement operator (obj1= --obj)*

#include<iostream>

using namespace std;

class Demo{

    public:

        int count = 100;

        int operator -- (){

            --count;

        }

        void display(){

            cout << "Count : " << count << endl;

        }

};

int main()

{

    Demo d1;

    d1.display();

    --d1;

    d1.display();

    return 0;

}

Q5

**CODE:**

*// 5.   Write a program to overload post decrement operator (obj1=obj--)*

#include<iostream>

using namespace std;

class Demo{

    public:

        int count = 100;

        int operator -- (int){

            --count;

        }

        void display(){

            cout << "Count : " << count << endl;

        }

};

int main()

{

    Demo d1;

    d1.display();

    d1--;

    d1.display();

    return 0;

}

Q6

**CODE:**

*// 6.   Write a program to overload Binary \*(multiply) operator for object of same class*

*//   Ex. class student s1,s2,s3     s3=s1\*s2*

#include<iostream>

using namespace std;

class Student{

    public:

        int m1 = 0, m2 = 0, m3 = 0, m4 = 0, m5 = 0;

        Student(int m1, int m2, int m3, int m4, int m5){

*this*->m1 = m1;

*this*->m2 = m2;

*this*->m3 = m3;

*this*->m4 = m4;

*this*->m5 = m5;

        }

        Student operator\*(Student s){

            Student temp(0, 0, 0, 0, 0);

            temp.m1 = *this*->m1 \* s.m1;

            temp.m2 = *this*->m2 \* s.m2;

            temp.m3 = *this*->m3 \* s.m3;

            temp.m4 = *this*->m4 \* s.m4;

            temp.m5 = *this*->m5 \* s.m5;

            return temp;

        }

        void display(){

            cout << "m1 = " << m1 << endl;

            cout << "m2 = " << m2 << endl;

            cout << "m3 = " << m3 << endl;

            cout << "m4 = " << m4 << endl;

            cout << "m5 = " << m5 << endl;

        }

};

int main(){

    Student s1(1, 2, 3, 4, 5), s2(6, 7, 8, 9, 10), s3(0,0,0,0,0);

    s3 = s1 \* s2;

    s3.display();

    return 0;

}

Q7

**CODE:**

*// 7.   Write a program to overload Binary \* (multiply) operator for object of different class*

#include<iostream>

using namespace std;

class PT;

class UT{

    public:

        int m1, m2, m3;

        UT(int m1, int m2, int m3){

*this*->m1 = m1;

*this*->m2 = m2;

*this*->m3 = m3;

        }

        friend UT operator + (PT a);

        void display(){

            cout << "m1 = " << m1 << endl;

            cout << "m2 = " << m2 << endl;

            cout << "m3 = " << m3 << endl << endl;

        }

};

class PT{

    public:

        int m1, m2, m3;

        PT(int m1, int m2, int m3){

*this*->m1 = m1;

*this*->m2 = m2;

*this*->m3 = m3;

        }

        void display(){

            cout << "m1 = " << m1 << endl;

            cout << "m2 = " << m2 << endl;

            cout << "m3 = " << m3 << endl;

        }

};

UT operator + (UT a, PT b){

    UT temp(0, 0, 0);

    temp.m1 = a.m1 + b.m1;

    temp.m2 = a.m2 + b.m2;

    temp.m3 = a.m3 + b.m3;

    return temp;

}

int main()

{

    UT u1(1, 2, 3), u2(4, 5, 6);

    PT p1(7, 8, 9), p2(10, 11, 12);

    UT result1 = u1 + p1;

    result1.display();

    UT result2 = u2 + p2;

    result2.display();

    return 0;

}

Q8

**CODE:**

*// 8.   Write program to overload = = operator to compare two object of student class*

#include<iostream>

using namespace std;

class Student{

    public:

        int m1, m2, m3;

        Student(int m1, int m2, int m3){

*this*->m1 = m1;

*this*->m2 = m2;

*this*->m3 = m3;

        }

        void operator==(Student s){

            if(*this*->m1 == s.m1 && *this*->m2 == s.m2 && *this*->m3 == s.m3){

                cout << "Same marks" << endl;

            }

            else{

                cout << "Different Marks" << endl;

            }

        }

};

int main()

{

    Student s1(1, 2, 3), s2(1, 2, 3), s3(4, 5, 6);

    s1 == s2;

    s1 == s3;

    return 0;

}

Q9

**CODE:**

*// 9.   Write program to overload > operator and find greater object for above program.*

#include<iostream>

using namespace std;

class Demo{

    public:

        int num;

        Demo(int num){

*this*->num = num;

        }

        void operator>(Demo d){

            if(*this*->num > d.num){

                cout << "Greater" << endl;

            }

            else{

                cout << "Smaller" << endl;

            }

        }

};

int main()

{

    Demo d1(199), d2(70);

    d1 > d2;

    return 0;

}