INDEX

S. No.	Program Name	Page No.	Date	Signature
8.	Program to demonstrate the concept of destructor.	13	30/03/22	
9.	Create the class TIME to store time in hours and minutes. Write a friend function to add two TIME objects.	14	06/04/22	
10.	Create two classes DM and DB. DM stores the distance in meters and centimeters and DB stores the distance in feet and inches. Write a program to add object of DM with the object of DB class.	15	13/04/22	
11.	Program to overload unary operator.	17	20/04/22	
12.	Program to overload binary operator.	18	27/04/22	
13.	Program to show multiple inheritance.	19	04/05/22	
14.	Program to show Multi-Level Inheritance.	20	04/05/22	
15.	Program to show hybrid inheritance.	21	11/05/22	
16.	Program to show run time polymorphism using virtual function.	22	18/05/22	

Program to demonstrate the concept of destructor.

Code

```
1
    // Program to demonstrate the concept of destructor
 2
 3
   #include <iostream>
4
    using namespace std;
 5
   class Point
7
    private:
8
       int x, y;
10
11
    public:
12
13
        // Constructor
        Point(int a, int b)
14
15
16
            x = a;
17
            y = b;
18
        }
19
        ~Point()
20
21
            cout<<"x: "<<x<<endl;
22
            cout<<"y: "<<y<<endl;
23
24
        }
25
    };
27
    int main()
28
29
        Point p(1, 3);
        return 0;
30
31
   }
```

Output

```
Loading personal and system profiles took 1887ms.

(base) PS D:\CUH\4th Sem\00P with C++\Lab\ cd "d:\CUH\4th Sem\00P with C++\Lab\11. Destructor\" ; if ($?) { g++ Destructor.cpp -o Destructor } ; if ($?) { .\Destructor } x: 1 y: 3
```

 $\frac{https://github.com/RPChinhara/OOP_Assignments/tree/master/11.\%20Destruct}{or}$

Create the class TIME to store time in hours and minutes. Write a friend function to add two TIME objects.

<u>Code</u>

```
1 // Write a class TIME to store time in hours and minutes. Write a friend function to add two TIME objects.
4 using namespace std;
8 private:
       int hours, minutes;
10
11 public:
12
     void input(int h, int min)
13
      hours = h;
14
15
          minutes = min;
16
17
18
       friend Time Add(Time, Time);
19
      void display()
20
21
           cout<<"Time: "<<hours<<":"<<minutes<<endl;
22
     }
23
24 };
25
26 Time Add(Time obj1, Time obj2)
27
28
      Time obj3;
     obj3.hours = obj1.hours + obj2.hours;
29
30
       obj3.minutes = obj1.minutes + obj2.minutes;
31
      return obj3;
32
33 }
35 int main()
36
       Time t1, t2, t3;
38
      t1.input(10, 36);
39
     t2.input(3, 13);
      t3 = Add(t1, t2);
41
42
43
      t3.display();
44
45
       return 0;
46 }
```

Output

(base) PS D:\CUH\4th Sem\00P with C++\Lab\11. Destructor> cd "d:\CUH\4th Sem\00P with C++\Lab\7. Time\" ; if (\$?) { g++ Time.cpp -o Time } ; if (\$?) { .\Time } Time: 13:49

https://github.com/RPChinhara/OOP_Assignments/tree/master/7.%20Time

Create two classes DM and DB. DM stores the distance in meters and centimeters and DB stores the distance in feet and inches. Write a program to add object of DM with the object of DB class.

Code

```
// Create two classes DM and DB. DM stores the distance in meters and centimeters a
3 #include <iostream>
4 using namespace std;
 6 class DM
7 {
8 public:
      int meters;
9
      int centimeters;
10
      DM(int m, int cm)
13
14
          meters = m;
15
          centimeters = cm;
16
         while (centimeters > 99)
19
              centimeters -= 100;
20
              meters++;
           }
21
22
      void display()
25
          cout<<"Meters: "<<meters<<endl;
26
27
          cout<<"Centimeters: "<<centimeters<<endl;
29 };
30
31 class DB
32 {
33 public:
     int feet;
34
35
       int inches;
36
      DB(int ft, int in)
37
38
           feet = ft;
          inches = in;
41
           while (inches > 11)
42
43
44
               inches -= 12;
45
               feet++;
          }
47
      }
```

```
49
        void display()
50
        {
             cout<<"Feet: "<<feet<<endl;
51
             cout<<"Inches: "<<inches<<endl;
52
         }
54
    };
55
    DM add_DB_to_DM(DM a, DB b)
57
58
        DM result(a.meters, a.centimeters);
60
        result.meters += 0.3048*b.feet;
61
         result.centimeters += 2.54*b.inches;
62
63
        while (result.centimeters > 99)
64
65
             result.centimeters -= 100;
             result.meters++;
66
67
         }
68
69
        return result;
70
71
72
    int main()
73
74
        DM obj1(8, 54);
        DB obj2(5, 6);
75
76
77
        obj1.display();
78
        obj2.display();
79
        DM result = add_DB_to_DM(obj1, obj2);
80
        result.display();
81
82
83
        return 0;
84
```

Output

```
Meters: 8
Centimeters: 54
Feet: 5
Inches: 6
Meters: 9
Centimeters: 69
```

 $\frac{https://github.com/RPChinhara/OOP_Assignments/tree/master/8.\%20Distance\\ \%20Conversion}{}$

Program to overload unary operator.

Code

```
// Program to overload unary operator
    #include <iostream>
    using namespace std;
    class Distance
8
    private:
       int meters, centimeters;
    public:
11
12
       Distance()
       meters = 0;
14
15
           centimeters = 0;
17
       Distance(int m, int cm)
18
20
            meters = m:
21
           centimeters = cm;
23
       // Displays Distance
      void display()
26
            cout<<meters<<" m "<<centimeters<<" cm"<<endl;
28
29
      Distance operator- ()
31
32
          meters = -meters;
           centimeters = -centimeters;
           return Distance(meters, centimeters);
   };
37
38
    int main()
40
41
       Distance d1(11, 10), d2(5, 1);
       -d1;
43
44
      d1.display();
45
       -d2;
46
47
      d2.display();
49
       return 0;
50 }
```

<u>Output</u>

```
(base) PS D:\CUH\4th Sem\00P with C++\Lab\8. Distance Conversion> cd "d:
\CUH\4th Sem\00P with C++\Lab\16. Overload Unary Operator\" ; if ($?) {
g++ UnaryOperator.cpp -o UnaryOperator } ; if ($?) { .\UnaryOperator }
-11 m -10 cm
-5 m -1 cm
```

https://github.com/RPChinhara/OOP Assignments/tree/master/16.%20Overload%20Unary%20Operator

Program to overload binary operator.

<u>Code</u>

```
// Program to overload binary operator
    #include <iostream>
    using namespace std;
    class Distance
8 private:
       int meters, centimeters;
10
11
    public:
      Distance()
12
            meters = 0;
15
    }
           centimeters = 0;
16
    Distance(int m, int cm)
18
            meters = m;
21
            centimeters = cm;
22
23
       // Displays Distance
       void display()
            cout<<meters<<" m "<<centimeters<<" cm"<<endl;</pre>
27
      Distance operator+ (Distance const &obj)
30
          Distance result;
33
            result.meters = meters + obj.meters;
            result.centimeters = centimeters + obj.centimeters;
            return result;
     }
38
39 };
   int main()
41
42
     Distance d1(10, 5), d2(2, 4);
Distance d3 = d1 + d2;
44
45
       d3.display();
47
        return 0;
48 }
```

Output

```
(base) PS D:\CUH\4th Sem\OUP with C++\Lab\16. Overload Unary Operator> cd "d:\CUH\4th Sem\OUP with C++\Lab\17. Overload Binary Operator\"; if ($?) { g++ BinaryOperator.cpp -o BinaryOperator}; if ($?) { .\BinaryOperator} } 12 m 9 cm
```

https://github.com/RPChinhara/OOP Assignments/tree/master/17.%20Overload%20Binary%20Operator

Program to show multiple inheritance.

Code

```
#include <iostream>
 4
    using namespace std;
 5
 6
    class A
    {
 8
    public:
       A()
 9
10
            cout<<"Constructor of A\n";
11
12
        }
13
    };
15
    class B
16
17
   public:
18
       B()
19
            cout<<"Constructor of B\n";
20
21
        }
22
    };
23
24
   class C: public B, public A
25
26
   public:
27
       C()
28
            cout << "Constructor of C\n";</pre>
29
30
31
    };
32
33
   int main()
34
    {
35
        Cc;
        return 0;
37
```

Output

```
(base) PS D:\CUH\4th Sem\00P with C++\Lab\17. Overload Binary Operator> cd "d:\CUH\4th Sem\00P with C++\Lab
\12. Multiple Inheritance\" ; if ($?) { g++ multipleInheritance.cpp -o multipleInheritance } ; if ($?) { .\
multipleInheritance }
Constructor of B
Constructor of A
Constructor of C
```

https://github.com/RPChinhara/OOP Assignments/tree/master/12.%20Multip le%20Inheritance

Program to show Multi-Level Inheritance.

Code

```
3 #include <iostream>
    using namespace std;
 5
 6
    class A
 7
 8
    public:
 9
        void FunctionA()
10
11
            cout<<"This is in class A";
12
         }
13
    };
14
15
    // Derived class (child)
    class Child: public A
16
17
    {
18
19
20
    // Derived class (grandchild)
21
22
    class GrandChild: public Child
23
    {
24
    };
25
    int main()
26
27
         GrandChild myObj;
28
         myObj.FunctionA();
29
        return 0;
30
31
```

Output

```
(base) PS D:\CUH\4th Sem\00P with C++\Lab\12. Multiple Inheritance> cd "d:\CUH\4th Sem\00P with C++\Lab\13.
Multi-level Inheritance\" ; if ($?) { g++ multiLevelInheritance.cpp -o multiLevelInheritance } ; if ($?) {
    \multiLevelInheritance }
This is in class A
```

https://github.com/RPChinhara/OOP Assignments/tree/master/13.%20Multi-level%20Inheritance

Program to show hybrid inheritance.

Code

```
#include <iostream>
    using namespace std;
 6
    class A
8
    public:
      A()
10
      {
11
           cout<<"This is constructor A\n";</pre>
12
13 };
14
15
    class B: public A
16
17 public:
     B()
18
19
           cout<<"This is constructor B\n";</pre>
20
21
22 };
23
   class C
24
26 public:
27
      void display()
28
           cout<<"This is member function of C\n";
     }
30
31
32
33 class D: public B, public C
34 {
    public:
36
       D()
           cout<<"Hybrid class D\n";
38
39
40
    };
42
    int main()
        D d;
44
      d.display();
46
        return 0;
48
```

Output

```
(base) PS D:\CUH\4th Sem\00P with C++\Lab\13. Multi-level Inheritance> cd "d:\CUH\4th Sem\00P with C++\Lab\
14. Hybrid Inheritance\" ; if ($?) { g++ hybridInheritance.cpp -o hybridInheritance } ; if ($?) { .\hybridI
nheritance }
This is constructor A
This is constructor B
Hybrid class D
This is member function of C
```

https://github.com/RPChinhara/OOP_Assignments/tree/master/14.%20Hybrid%20Inheritance

Program to show run time polymorphism using virtual function.

Code

```
#include <iostream>
    using namespace std;
    class Shape
6
8
    public:
           virtual void calculate()
11
                   cout<<"Area of your Shape ";
12
13 };
14
15 class Rectangle : public Shape
16
17 public:
           int width, height, area;
18
19
20
          void calculate()
21
                   cout<<"Enter Width of Rectangle: ";
22
23
                   cin>>width;
24
25
                   cout<<"Enter Height of Rectangle: ";
26
                   cin>>height;
28
                   area = height * width;
                   cout << "Area of Rectangle: " << area << "\n";
29
          }
30
31 };
33
    class Square : public Shape {
           int side, area:
35
36
37
           void calculate()
38
                   cout << "Enter one side your of Square: ";
40
                   cin >> side;
41
                   area = side * side;
42
                   cout << "Area of Square: " << area;
43
44
45 };
46
    int main()
47
48
49
       Rectangle r;
50
           r.calculate();
51
      Square sq;
52
          sq.calculate();
53
54
55
           return 0;
```

Output

```
Enter Width of Rectangle: 10
Enter Height of Rectangle: 20
Area of Rectangle: 200
Enter one side your of Square: 23
Area of Square: 529
```

https://github.com/RPChinhara/OOP Assignments/tree/master/18.%20Polymorphism