C++ Assignment [16-01-2018]

Emp Name	Programs And Output

1_Divya _Bolu	**/* Has a Relationship Program */**
	
	#include <iostream>
	
	class base{
	    public:
	        void f(){
	            std::cout <<"in base class"<<std::endl;
	        }
	};
	
	class derived:public base{
	    public:
	        derived()
	        {
	            base obj;
	            obj.f();
	        }
	
	};
	
	int main(){
	    derived obj1;
	
	    return 0;
	}
	-- 	
	Output:
	in base class

2_Arjun	**/* Accessing from main() public member of base class, Base class is derived private */**
	
	#include<iostream>
	using namespace std;
	class Baseclass
	{
	public:
	        int val;
	        void display()
	        {
	                cout<<"enter value in parent class:";
	                cin>>val;
	                cout<<"In parent class "<<val<<endl;
	        }
	};
	class Derivedclass1:private Baseclass

```
{
public:
        int val2;
        Derivedclass1()
        {
        cout<<"Derived class constructor"<<endl;
        Baseclass::display();
        }
};
int main()
{
        Derivedclass1 b1;

}
```

Output:

```
Derived class constructor
enter value in parent class:4
In parent class 4
```

3_Deepika

/* Various combinations of default arguements in c++ functon. */

```
#include<iostream>
using namespace std;
class base
{
        public:
                int sum(int val1,int val2,int val3=0,int val4=0)
                {
                        int res;
                        res=val1+val2+val3+val4;
                        return res;
                }
};
int main()
{
        base obj;
        cout << obj.sum(10,10)<<endl;
        cout << obj.sum(10,10,10)<<endl;
        cout << obj.sum(10,10,10,10)<<endl;
        return 0;
}
```

Output:

```
20
30
40
```

4_Anan_Mishra

/* User Implementation of Namespace*/

```
#include "header.h"
#include <string>
#include <iostream>

int main()
{
   using namespace Test;
   using namespace std;

   string s = Func();
   std::cout << s << std::endl; // "Hello from new"
   return 0;
}
```

-----------------------------Header file---------------------------------------

```
#include <string>

namespace Test
{
   namespace old_ns
   {
      std::string Func() { return std::string("Hello from old"); }
   }

   inline namespace new_ns
   {
      std::string Func() { return std::string("Hello from new"); }
   }
}
```

--

Output:

Hello from new

5_Sai_Krishna

Output:

6_Ashish_Jain

/*Overload Constructor*/

```
#include <iostream>
using namespace std;

class construct
{
public:
   float area;
   construct()
```

```
    {
        area = 0;
    }
    construct(int num1, int num2)
    {
        area = num1 * num2;
    }

    void disp()
    {
        cout<< area<< endl;
    }
};

int main()
{
    construct obj;
    construct obj2( 10, 20);

    obj.disp();
    obj2.disp();
    return 1;
}
```

Output:

```
0
200
```

7_Rathod_Raja

/* Shallow Copy Program */

```
#include<iostream>
using namespace std;

class shallow_copy
{
            int data1,data2;
            public:

            void setdata(int value1, int value2)
            {
                        data1=value1;
                        data2=value2;
            }

            void showdata()
            {
                        cout << "data1="<<data1<<"\ndata2="<<data2<<"\n";
            }

};
```

```
 int main()
{
           shallow_copy obj1;
           obj1.setdata(10,20);
           shallow_copy obj2;
           obj2=obj1;
           obj2.showdata();

return 0;
}
```

Output:

```
data1=10
data2=20
```

/* Deep Copy Program */

```
#include<iostream>
using namespace std;

class deep_copy
{
           int data1,data2,*ptr;
           public:

           deep_copy()
           {
                      ptr=new int ;
           }
           void setdata(int value1, int value2,int value3)
           {
                      data1=value1;
                      data2=value2;
                      *ptr=value3;
           }
           void showdata()
           {
                      cout << "data1="<<data1<<"\ndata2="<<data2<<"\nptr="<<*ptr<<"\n";
           }
           deep_copy(deep_copy &ref)
           {
                      this->data1=ref.data1;
                      this->data2=ref.data2;
                      this->ptr=new  int;
                      *ptr=*(ref.ptr);
           }
           ~deep_copy()
           {
                      delete ptr;
```

```
        }
};

int main()
{
        deep_copy obj1;
        obj1.setdata(10,20,30);
        deep_copy obj2=obj1;
        obj1.showdata();
return 0;
}
```

Output:

```
data1=10
data2=20
ptr=30
```

8_Harish

/*Program to Use of Ststic Memebr */

```
#include <cstdlib>
#include <iostream>

using namespace std;

class Box
{
  public:
    static int objectCount;
    // Constructor definition
    Box(double l=2.0, double b=2.0, double h=2.0)
    {
      cout <<"Constructor called." << endl;
      length = l;
      breadth = b;
      height = h;
      // Increase every time object is created
      objectCount++;
    }
    double Volume()
    {
      return length * breadth * height;
    }
    static int getCount()
    {
      return objectCount;
    }

private:
    double length;     // Length of a box
    double breadth;   // Breadth of a box
    double height;    // Height of a box
};

// Initialize static member of class Box
int Box::objectCount = 0;

int main(void)
{

  // Print total number of objects before creating object.
  cout << "Inital Stage Count: " << Box::getCount() << endl;

  Box Box1(3.3, 1.2, 1.5);    // Declare box1
  Box Box2(8.5, 6.0, 2.0);    // Declare box2

  // Print total number of objects after creating object.
  cout << "Final Stage Count: " << Box::getCount() << endl;

  return 0;
}
```

Output:

```
Inital Stage Count: 0
Constructor called.
Constructor called.
Final Stage Count: 2
```

9_Uday

/* Change the value of constant member function using Mutable */

```
#include <iostream>
using namespace std;

class Sample
{
        int x;
        mutable int y;

        public:
        Sample(int a=0, int b=0)
        { x=a; y=b;}

        //function to set value of x
        void setx(int a=0)
        {x = a;}

        //function to set value of y
        //value of y being changed, even if member function is constant.
        void sety(int b=0) const
        {y = b;}
```

```
            //function to display x and y.
            //this has to be const type, if member function is constant type.
            void display() const
            {
                        cout<<endl<<"x: "<<x<<" y: "<<y<<endl;
            }
};
```

Output:

```
Value before change:
x: 10 y: 20

Value after change:
x: 10 y: 200
```

10_Sandeep_R

/* Program for to demonstrate viratual destructor */

```
#include<iostream>
using namespace std;
class base
{
        int val;
public:
        base()
        {
                cout<<"base class constructor"<<endl;
        }
       ~base()
        {
                cout<<"base class destructor"<<endl;
        }
};
class derived1:public base
{
public:
        derived1()
        {
                cout<<"derived1 class constructor"<<endl;
        }
        ~derived1()
        {
                cout<<"derived1 class destructor"<<endl;
        }
};

int main()
{
```

	`derived1 *obj1=new derived1();`
	`base *obj2 = obj1;`
	`delete obj2;`
	`}`
	-- 	
	Output:
	
	base class constructor
	derived1 class constructor
	base class destructor

11_Harnath	**/* Object Slicing */**
	
	`using namespace std;`
	
	`class Base`
	`{`
	`public:`
	`Base(int val)`
	`{`
	`val_ = val;`
	`}`
	
	`void print()`
	`{`
	`cout<< "In Base::print() : val_ " << val_ <<endl;`
	`}`
	`private:`
	`int val_;`
	`};`
	`class Derived : public Base`
	`{`
	`public:`
	`Derived(int val, int b):Base(val)`
	`{`
	`b_ = b;`
	`}`
	
	`void print()`
	`{`
	`cout<< "In Derived::print() : b_ " << b_ <<endl;`
	`}`
	`private:`
	`int b_;`
	`};`
	
	`void disp (Base ob)`
	`{`
	`ob.print();`
	`}`

```
int main()
{
            Base b(10);
            Derived d(15, 25);
            disp(b);
            disp(d); // slicing will happen
            return 0;
}
```

Output:

```
In Base::print() : val_ 10
In Base::print() : val_ 15
```

12_Smruti_Ranjan

/* Friends function */

```
#include <iostream>
using namespace std;

class Distance
{
    private:
        int meter;
    public:
        Distance(): meter() { }

        friend int addFive(Distance); //friend function
};

// friend function definition
int addFive(Distance d)
{
    //accessing private data from non-member function
    d.meter += 5;
    return d.meter;
}

int main()
{
    Distance D;
    cout<<"Distance: "<< addFive(D);
    return 0;
}
```

Output:

```
Distance: 5
```

13_Ishaque	**/* Multiple object passing to single object using this Pointer */**

```
#include<iostream>
using namespace std;

class student
{
    char name[100];
    int age,roll;
    float percent;
    public:
        void getdata()
        {
            cout<<"Enter data"<<endl;
            cout<<"Name:";
            cin>>name;
            cout<<"Age:";
            cin>>age;
            cout<<"Roll:";
            cin>>roll;
            cout<<"Percent:";
            cin>>percent;
            cout<<endl;
        }
        student & max(student &s1,student &s2)
        {
            if(percent>s1.percent && percent>s2.percent)
                return *this;
            else if(s1.percent>percent && s1.percent>s2.percent)
                return s1;
            else if(s2.percent>percent && s2.percent>s1.percent)
                return s2;
        }
        void display()
        {
            cout<<"Name:"<<name<<endl;
            cout<<"Age:"<<age<<endl;
            cout<<"Roll:"<<roll<<endl;
            cout<<"Percent:"<<percent;
        }
};

int main()
{
    student s,s1,s2,s3;
    s1.getdata();
    s2.getdata();
    s3.getdata();
    s=s3.max(s1,s2);
    cout<<"Student with highest percentage"<<endl;
    s.display();
    return 0;
}
```

Output:

```
Enter data
Name:ram
Age:11
Roll:1
Percent:80

Enter data
Name:sham
Age:12
Roll:4
Percent:90

Enter data
Name:lakhan
Age:13
Roll:7
Percent:99

Student with highest percentage
Name:lakhan
Age:13
Roll:7
Percent:99
```