C++ Assignment [11-01-2018]

Emp Name	Program And Output

1_Divya_P	**/* Checking the access specifier */**
	
	#include<iostream>
	using namespace std;
	
	class Sample
	{
	private:
	int pri=10;
	
	public:
	int pub = 100;
	
	protected:
	int prot = 1000;
	public:
	void display()
	{
	cout<<"private = "<<pri<<endl<<"publiv val = "<<pub<<endl<<"protected value="<<prot<<endl;
	}
	
	};
	
	int main()
	{
	Sample s1;
	s1.display();
	
	return 0;
	}
	
	--- 	
	Output:
	private = 10
	publiv val = 100
	protected value=1000

2_Divya_Bolu	**/* Get and Set Interface to variable */** #include<iostream> using namespace std; class Sample { int val; public: int setval(void)

```
                {
                        cout<<"set the value:";
                        cin>>val;
                }
        int getval(void)
                {
                        cout<<"value is:"<<val<<endl;
                }
};
int main()
{
        Sample obj;
        obj.setval();
        obj.getval();
}
```

Output:

set the value:4

value is:4

3_Harish

```
/* Operator Overloading */

#include <cstdlib>
#include<iostream>

using namespace std;

class Complex {
private:
   int real, imag;
public:
   Complex(int r = 0, int i =0)  {real = r;   imag = i;}

   // This is automatically called when '+' is used with
   // between two Complex objects
   Complex operator + (Complex const &obj) {
       Complex res;
       res.real = real + obj.real;
       res.imag = imag + obj.imag;
       return res;
   }
   Complex operator - (Complex const &obj);
   void print() { cout << real << " + i" << imag << endl; }
};

Complex Complex ::operator - (Complex const &obj)
{
        Complex res1;
        res1.real = real - obj.real;
        res1.imag = imag - obj.imag;
```

	return res1;
	}
	
	int main()
	{
	int result = 0 ;
	Complex c1(10, 5), c2(2, 4),c4(0,0);
	Complex c3 = c1 + c2; // An example call to "operator+"
	c4 = c1- c2;
	c3.print();
	c4.print();
	return result;
	
	}
	--- 	
	Output:
	
	12 + i9
	8 + i1

4_Dayanand	**/* Function Overloading */**
	
	#include<iostream>
	using namespace std;
	
	int volume(int);
	double volume(double,int);
	long volume(long,int,int);
	
	int main()
	{
	cout << " Calling the volume() function for computing the volume of a cube -" << volume(10) << endl;
	cout << " Calling the volume() function for computing the volume of a cylinder -" << volume(2.5,8) << endl;
	cout << " Calling the volume() function for computing the volume of a rectengular box -" << volume(100L,75,15) << endl;
	return 0;
	}
	
	int volume(int side) //cube
	{
	int res;
	res=side*side*side;
	return res;
	}
	double volume(double radius, int hight) //cylinder
	{
	double res;

```
        res=3.14519*radius*radius*hight ;
        return res;
}
long volume(long len,int width,int hight) //rectengular box
{
        long res;
        res=len*width*hight;
        return res;
}
```

Output:

```
Calling the volume() function for computing the volume of a cube -1000
Calling the volume() function for computing the volume of a cylinder -157.26
Calling the volume() function for computing the volume of a rectengular box -112500
```

5_Anan

```
/* Constructor and Destructor */

#include <iostream>
using namespace std;

class Baseclass
{
public:
        int a,b;

    Baseclass()
    {
      cout << "setting value in constructor."<<endl;
      a=10;
      b=20;
    }
    ~Baseclass()
    {
      cout<< "\nprinting in destructor."<<endl;
    }
};

int main()
{
    Baseclass c;

    cout << "in main." << endl;
    cout << "a: "<< c.a << endl << "b: "<< c.b;
    cout << "\npress enter to exit."<<endl;
    cin.get();
    cout << "exiting main."<< endl;

    return 0;
}
```

	--- **Output:** setting value in constructor. in main. a: 10 b: 20 press enter to exit. exiting main. printing in destructor.

6_Mallikarjun	**/* Sample class member with scope resolution */**
	
	#include<iostream>
	using namespace std;
	class Sample
	{
	int val;
	public:
	int setval(void);
	int getval(void);
	};
	inline int Sample::setval(void)
	{
	cout<<"set the value:";
	cin>>val;
	}
	inline int Sample::getval(void)
	{
	cout<<"value is:"<<val<<endl;
	}
	int main()
	{
	Sample obj;
	obj.setval();
	obj.getval();
	}
	
	--- 	
	
	Output:
	set the value:12
	value is:12

7_Ashish	**/* Private inheritance */** #include <iostream> using namespace std;

	class base
	{
	  private:
	  int priv;
	  protected:
	  int prot;
	  public:
	  void setVal(int value)
	  {
	  prot=value;
	  }
	};
	
	class derived:private base
	{
	  public:
	  void printVal(void)
	  {
	  setVal(10);
	  cout << "value of prot: " << prot << endl;
	  }
	};
	
	int main()
	{
	  derived objderived ;
	  objderived .printVal();
	  return 0;
	}
	
	--- 	
	Output:
	
	`Compile time error`
	In, private inheritance only protected data member and member functions can be accessed by the derived class

8_Uday	**/* Inheritance Public */**
	
	#include<iostream>
	using namespace std;
	
	//Base class
	
	class Parent
	{
	  public:
	  int id_p;
	};

	// Sub class inheriting from Base Class(Parent)
	
	class Child : public Parent
	{
	public:
	int id_c;
	};
	
	//main function
	
	int main()
	{
	
	Child obj1;
	
	// An object of class child has all data members
	// and member functions of class parent
	obj1.id_c = 7;
	obj1.id_p = 91;
	cout << "Child id is " << obj1.id_c << endl;
	cout << "Parent id is " << obj1.id_p << endl;
	
	return 0;
	}
	--- 	
	Output:
	Child id is 7
	Parent id is 91

9_Srinivas	/***Protected Inheritance** */
	
	#include <iostream>
	
	using namespace std;
	
	class GrandParent{
	public:
	void grandParentMethod(void){ cout<<"Method in the grand parent class"<<endl; }
	};
	
	class Parent : protected GrandParent{
	public:
	void parentMethod(void){ cout<<"Method in the parent class"<<endl; }
	};
	
	class Child: protected Parent{
	public:
	void
	childMethod(void){

	cout<<"Method in the child class"<<endl;
	parentMethod();
	grandParentMethod();
	}
	};
	
	int
	main(void){
	
	Child C;
	C.childMethod();
	return 0;
	
	}
	
	--- 	
	Output:
	Method in the child class
	Method in the parent class
	Method in the grand parent class

10_Deepika	/* **Constructor ,Destructor in base class and derived class.** */
	
	#include<iostream>
	using namespace std;
	class base1
	{
	public:
	base1()
	{
	cout<<"constructor of base1 class a\n";
	}
	~base1()
	{
	cout<<"destructor of base1 class a\n";
	}
	};
	class base2
	{
	public:
	base2()
	{
	cout<<"constructor of base2 class b\n";
	}
	~base2()
	{
	cout<<"destructor of base2 class b\n";
	}
	};

```
class derived :public base1,public base2//drived class
{
        public:
        derived()
        {
                cout<<"constructor of derived class c\n";
        }
        ~derived()
        {
                cout<<"destructor of derived class c\n";
        }
};
int main()
{
        derived obj;
        return 0;
}
```

Output:

constructor of base1 class a
constructor of base2 class b
constructor of derived class c
destructor of derived class c
destructor of base2 class b
destructor of base1 class a

11_Sandeep_A

/* Constructor and Destructor with derived class [Multilevel] */

```
#include<iostream>
using namespace std;
class Baseclass1
{
public:
      int val;
      void display()
      {
            cout<<"enter value in base class1:";
            cin>>val;
            cout<<"In Base class1 "<<val<<endl;
      }
};
class Baseclass2
{
      public:
      int val2;
      void Baseclass2_display()
      {
            cout<<"enter value in Base class2:";
```

```
            cin>>val2;
            cout<<"In Base class2 value"<<val2<<endl;
        }
};
class Derivedclass:public Baseclass1,public Baseclass2
{
};
int main()
{
        Derivedclass DC;
        DC.display();
        DC.Baseclass2_display();
}
```

Output:

```
enter value in base class1:12
In Base class1 12
enter value in Base class2:13
In Base class2 value13
```

12_Ramya_B

```
/* Constructor and Destructor with derived class [Multipal] */

#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:public Baseclass
{
        public:
        int val2;
        void childdisplay()
        {
                cout<<"enter value in child class1:";
                cin>>val2;
                cout<<"In child1 class value"<<val2<<endl;
        }
};
class Derivedclass2:public Derivedclass1
```

	{
	};
	int main()
	{
	Derivedclass1 DC1;
	/* Accesing Parent class member in child1*/
	DC1.display();
	Derivedclass2 DC2;
	/*Accessing parent and child class members in child2*/
	DC2.display();
	DC2.childdisplay();
	}
	
	--- 	
	Output:
	enter value in parent class:10
	In parent class 10
	enter value in parent class:10
	In parent class 10
	enter value in child class1:20
	In child1 class value20

13_Sweta	**/*Constructor & Destructor with Hierarchical */**
	
	#include <iostream>
	using namespace std;
	
	class Number
	{
	private:
	int num;
	public:
	void getNumber(void)
	{
	cout << "Enter an integer number: ";
	cin >> num;
	}
	//to return num
	int returnNumber(void)
	{ return num; }
	};
	
	//Base Class 1, to calculate square of a number
	class Square:public Number
	{
	public:
	int getSquare(void)
	{
	int num,sqr;
	num=returnNumber(); //get number from class Number

	sqr=num*num;
	return sqr;
	}
	};
	
	//Base Class 2, to calculate cube of a number
	class Cube:public Number
	{
	private:
	
	public:
	int getCube(void)
	{
	int num,cube;
	num=returnNumber(); //get number from class Number
	cube=num*num*num;
	return cube;
	}
	};
	int main()
	{
	Square objS;
	Cube objC;
	int sqr,cube;
	
	objS.getNumber();
	sqr =objS.getSquare();
	cout << "Square of "<< objS.returnNumber() << " is: " << sqr << endl;
	
	objC.getNumber();
	cube=objC.getCube();
	cout << "Cube of "<< objS.returnNumber() << " is: " << cube << endl;
	
	return 0;
	}
	--- 	
	Output:
	
	Enter an integer number: 5
	Square of 5 is: 25
	Enter an integer number: 6
	Cube of 5 is: 216

14_Sweta	**/*Constructor & Destructor with Hybrid */** #include <iostream> using namespace std; class mm { protected: int rollno;

```
public:
void get_num(int a)
{ rollno = a; }
void put_num()
{ cout << "Roll Number Is:"<< rollno << "\n"; }
}; class marks : public mm
{
protected:
int sub1;
int sub2;
public:
void get_marks(int x,int y)
{
sub1 = x;
sub2 = y;
}
void put_marks(void)
{
cout << "Subject 1:" << sub1 << "\n";
cout << "Subject 2:" << sub2 << "\n";
}
};
class extra
{
protected:
float e;
public:
void get_extra(float s)
{e=s;}
void put_extra(void)
{ cout << "Extra Score::" << e << "\n";}
};
class res : public marks, public extra{
protected:
float tot;
public:
void disp(void)
{
tot = sub1+sub2+e;
put_num();
put_marks();
put_extra();
cout << "Total:"<< tot;
}
};
int main()
{
res std1;
std1.get_num(10);
std1.get_marks(10,20);
```

	std1.get_extra(33.12);
	std1.disp();
	return 0;
	}
	--- 	
	Output:
	Roll Number Is:10
	Subject 1:10
	Subject 2:20
	Extra Score::33.12
	Total:63.12

15_Ishaque	**/* Function Overriding */**
	
	#include<iostream>
	using namespace std;
	class Base
	{
	public:
	void func()
	{
	cout<<"i am in Base class"<<endl;
	}
	};
	class Derived : public Base
	{
	
	public:
	void func()
	{
	cout<<"i am in derived class"<<endl;
	}
	};
	int main()
	{
	Derived obj;
	obj.func();
	obj.Base::func();
	}
	--- 	
	Output:
	i am in derived class
	i am in Base class

16_Rahul	**/* Sample program to show the functionality of virtual function */** #include<iostream> using namespace std; class base

	{
	public:
	virtual void print ()
	{
	cout<< "print base class" <<endl;
	}
	void show ()
	{
	cout<< "show base class" <<endl;
	}
	};
	
	class derived:public base
	{
	public:
	void print ()
	{
	cout<< "print derived class" <<endl;
	}
	void show ()
	{
	cout<< "show derived class" <<endl;
	}
	};
	
	int main()
	{
	base *bptr;
	derived dobj;
	bptr = &dobj;
	
	//virtual function, binded at runtime
	bptr->print();
	
	// Non-virtual function, binded at compile time
	bptr->show();
	}
	
	---------- 	
	Output:
	
	print derived class
	show base class

17_Sai Krishna	**/* Pure virtual function */** #include <iostream> using namespace std;

```
// Abstract class
class Shape
{
   protected:
     float l;
   public:
     void getData()
     {
        cin >> l;
     }

     // virtual Function
     virtual float calculateArea() = 0;
};

class Square : public Shape
{
   public:
     float calculateArea()
     {   return l*l;  }
};

class Circle : public Shape
{
   public:
     float calculateArea()
     { return 3.14*l*l; }
};

int main()
{
   Square s;
   Circle c;

   cout << "Enter length to calculate the area of a square: ";
   s.getData();
   cout<<"Area of square: " << s.calculateArea();
   cout<<"\nEnter radius to calculate the area of a circle: ";
   c.getData();
   cout << "Area of circle: " << c.calculateArea();

   return 0;
}
```

Output:

Enter length to calculate the area of a square: 4
Area of square: 16
Enter radius to calculate the area of a circle: 5
Area of circle: 78.5