CPP programs

1. Develop a C++ program to illustrate scope resolution

```
#include<iostream>
using namespace std;
int a=100,b=500;
int main()
{
        int a=50;
        cout<<"local variable of a is : "<<a<<endl;
        cout<<"global variable of a is : "<<::a<<endl;
        cout<<"global variable of b is : "<<b<<endl;
        ::a=1000;
        cout<<"value of a is : "<<::a<<endl;
}
```

2. Develop a C++ program to illustrate namespaces.

```
#include<iostream>
using namespace std;
namespace one
{
int a=10;
}
namespace two
{
        double a=50.999;
}
namespace three
{
        long a=59999977;
}
namespace four
{
        float a=7.8f;
}
int main()
{
        cout<<one::a<<endl;
        cout<<two::a<<endl;
        cout<<three::a<<endl;
        cout<<four::a<<endl;

}
```

3. Develop a C++ program illustrating Inline Functions

```
#include<iostream>
using namespace std;
inline int show(int a, int b)
    {
        return (a>b?a:b);
    }
int main()
{
    int a,b;
    cout<<"enter a and b value";
    cin>>a>>b;
    int x=show(a,b);
    cout<<x;
}
```

4. Develop a C++ program demonstrating a Bank Account with necessary datamembers and member functions.
Or
Develop a C++ program for illustrating Access Specifiers :public and private.

```
#include<iostream>
using namespace std;
class bank
{
        private:
        string fname,lname;
        int alc, pin;
        public:
        void get()
        {
                cout<<"enter first name:"<<endl;
                cin>>fname;
                cout<<"enter last name:"<<endl;
                cin>>lname;
                cout<<"enter alc number and pin no."<<endl;
                cin>>alc>>pin;

                }
                void show()
                {
                        cout<<"first name="<<fname<<endl<<"last name="<<lname<<endl<<"alc
no="<<alc<<endl<<"pin no"<<pin<<endl;
                        }
```

```
};
int main()
{
        bank b;
    b.get();
        b.show();
}
```

5. Develop a C++ program to illustrate this pointer.

```
#include<iostream>
using namespace std;
class pointers
{
        private:
                string fname, lname;
                int pin, alc;
                public:
        void get()
        {
                cout<<"enter first name:"<<endl;
                cin>>this->fname;
                cout<<"enter last name:"<<endl;
                cin>>this->lname;
                cout<<"enter alc number and pin no."<<endl;
                cin>>this->alc;
                cin>>this->pin;

                }
                void show()
                {
        cout<<"first name="<<this->fname<<endl<<"last name="<<this->lname<<endl<<"alc
no="<<this->alc<<endl<<"pin no="<<this->pin<<endl;
                        }
};
int main()
{
        pointers p;
        p.get();
        p.show();
}
```

Or

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

```
{
	private:
		int length,breadth;
		public:
			void get(int l, int b)
			{
				this->length=l;
				this->breadth=b;
			}
			void show()
			{
				cout<<"area of box is "<<length*breadth;
			}
};
int main()
{
	box b;
	b.get(10,30);
	b.show();
	}
```

6. Develop a C++ program illustrate function overloading.

```
//function overloading
#include<iostream>
using namespace std;
class shape
{
	public:
		void area(int side)
		{
			cout<<"Area of square is"<<side*side<<endl;
		}
		void area(float r)
		{
			cout<<"Area of circle is"<<3.14*r*r<<endl;
		}
		void area(int l, int b)
		{
			cout<<"Area of rectangle is"<<l*b<<endl;
		}
};
int main()
{
	shape s;
	s.area(5);
	s.area(5.5f);
```

```
        s.area(10, 20);
}
```

7. Develop a C++ program to illustrate the use of default arguments.

```
#include<iostream>
using namespace std;
int interest(int p, int t, int r=5)
{
        return (p*t*r)/100;
}
int main()
{
        int p,t,r;
        cout<<"enter p,t,r"<<endl;
        cin>>p>>t>>r;
        cout<<interest(p,t)<<endl;
        cout<<interest(p,t,r)<<endl;
}
```

8. Develop a C++ program illustrating friend function.

```
#include<iostream>
using namespace std;
class arithematic
{
        private:
                int a,b;
                public:
                        void get()
                        {
                                cout<<"enter a and b"<<endl;
                                cin>>a>>b;
                        }
                        friend void arith(arithematic &s);
};
void arith(arithematic &s)
{
        s.get();
        cout<<"Addition ="<<s.a+s.b<<endl;
        cout<<"subtraction ="<<s.a-s.b<<endl;
        cout<<"multiplication ="<<s.a*s.b<<endl;
}
int main()
{
        arithematic s;
```

```
        arith(s);
}
```

9. **Develop a C++ Program to illustrate the use of Constructors and Destructors.**

```
#include<iostream>
using namespace std;
class volume
{
        private:
                int l,b,h;
                public:
                        volume()
                        {
                        cout<<"enter length, breadth and height"<<endl;
                        cin>>l>>b>>h;
                }
                void show()
                {
                        cout<<"volume of a box="<<l*b*h<<endl;
                }
                ~ volume()
                {
                        cout<<"destructor is involved"<<endl;
                }
};
int main()
{
        volume v;
        v.show();
}
```

10. Develop a C++ program illustrating Constructor overloading.

```
#include<iostream>
using namespace std;
class Rectangle
{
        private:
                int length, breadth;
                public:
                Rectangle()
                {
                        length=10;
                        breadth=20;
```

```
        }


        Rectangle(int x, int y)
        {
                length=x;
                breadth=y;
        }
        void area()
        {
                cout<<"Area of rectangle:"<<length*breadth<<endl;
        }
};
int main()
{
 Rectangle r1;
 Rectangle r2 (20,20);
 r1.area();
 r2.area();
}
```

11. Develop a C++ program illustrating Copy Constructor.

```
#include<iostream>
using namespace std;
class person
{
        private:
                string fname,lname;
                public:
                        person(string f, string l)
                        {
                         fname=f;
                         lname=l;
                        }
                        person(person &p1)
                        {
                                fname=p1.fname;
                                lname=p1.lname;
                        }
                        void show()
                        {
                                cout<<"first name= "<<fname<<endl;
                                cout<<"Last name= "<<lname<<endl;
                        }
};
int main()
```

```
{
        person p1 ("sanjit", "das");
        p1.show();
        person p2(p1);
        p2.show() ;
}
```

12) Develop a C++ program to Overload Unary, and Binary Operators using memberfunction.

Unary operator using member function

```
#include<iostream>
using namespace std;
class sample
{
        private:
                int a,b,c;
                public:
                        void get()
                        {
                                a=10;
                                b=20;
                                c=-30;
                        }
                        void show()
                        {
                                cout<<"a= "<<a<<endl;
                                cout<<"b= "<<b<<endl;
                                cout<<"c= "<<c<<endl;

                        }
                        void operator -()
                        {
                                a=-a;
                                b=-b;
                                c=-c;
                        }
};
int main()
{
        sample s;
        s.get();
        s.show();
        -s;
        cout<<"after uniary minus"<<endl;
        s.show();
}
```

Binary using member function

```
#include<iostream>
using namespace std;
class complex
{
        private:
                int real, imaginary;
                public:
                        void get()
                        {
                                cout<<"enter real number:"<<endl;
                                cin>>real;
                                cout<<"enter imaginary number:"<<endl;
                                cin>>imaginary;
                        }
                        void operator +(complex c2)
                        {
                                cout<<real+c2.real<<"+"<<imaginary+c2.imaginary<<"i";
                        }
};
int main()
{

complex c1, c2;
c1.get();
c2.get();
c1+c2;
}
```

13)Develop a C++ program to Overload Unary, and Binary Operators using friend function.

Uniary using friend function

```
#include<iostream>
using namespace std;
class sample
{
        private:
                int a,b,c;
                public:
                        void get()
                        {
                                a=10;
                                b=20;
                                c=-30;
```

```
                }
                void show()
                {
                        cout<<"a= "<<a<<endl;
                        cout<<"b= "<<b<<endl;
                        cout<<"c= "<<c<<endl;

                }
                friend void operator -(sample &s);
};
void operator -(sample &s)
{
        s.a=-s.a;
        s.b=-s.b;
        s.c=-s.c;
}
int main()
{
        sample s;
        s.get();
        s.show();
        -s;
        cout<<"after uniary minus"<<endl;
        s.show();
}
```

Binary using friend function

```
#include<iostream>
using namespace std;
class complex
{
        private:
                int real, imaginary;
                public:
                        void get()
                        {
                                cout<<"enter real number:"<<endl;
                                cin>>real;
                                cout<<"enter imaginary number:"<<endl;
                                cin>>imaginary;
                        }
                friend void operator +(complex &c1, complex &c2);
};
void operator +(complex &c1, complex &c2)
{
        cout<<c1.real+c2.real<<"+"<<c1.imaginary+c2.imaginary<<"i";
```

```
}
int main()
{
complex c1, c2;
c1.get();
c2.get();
c1+c2;
}
```

14) Develop C++ Programs to incorporate various forms of Inheritance

Single inheritance

```
#include<iostream>
using namespace std;
class student
{
        protected:
                string name, gender;
                int age;
                public:
                void gets()
                {
                        cout<<"enter name, gender, age of student"<<endl;
                        cin>>name>>gender>>age;
                }
                void shows()
                {
                        cout<<"Name= "<<name<<endl;
                        cout<<"gender= "<<gender<<endl;
                        cout<<"age= "<<age<<endl;
                        }
};
class mark: public student
{
        private:
                string reg;
                int m1,m2,m3,m4,m5;
                public:
                void getm()
                {
                        cout<<"enter reg no."<<endl;
                        cin>>reg;
                        cout<<"enter 5 subject marks"<<endl;
                        cin>>m1>>m2>>m3>>m4>>m5;
                }
                void show()
```

```
		{
			cout<<"reg= "<<reg<<endl;
			cout<<"marks="<<m1<<" "<<m2<<" "<<m3<<" "<<m4<<" "<<m5<<endl;
		}

};
int main()
{
	mark m;
	m.gets();
	m.getm();
	m.shows();
	m.show();

}
```

Multilevel inheritance

```
//multilevel inheritance
#include<iostream>
using namespace std;
class student
{
	protected:
		string name, gender;
		int age;
		public:
		void gets()
		{
			cout<<"enter name, gender, age of student"<<endl;
			cin>>name>>gender>>age;
		}
		void shows()
		{
			cout<<"Name= "<<name<<endl;
			cout<<"gender= "<<gender<<endl;
			cout<<"age= "<<age<<endl;
			}
};
class mark: public student
{
	protected:
		string reg;
		int m1,m2,m3,m4,m5;
		public:
		void getm()
		{
```

```
                cout<<"enter reg no."<<endl;
                cin>>reg;
                cout<<"enter 5 subject marks"<<endl;
                cin>>m1>>m2>>m3>>m4>>m5;
        }
        void show()
        {
                cout<<"reg= "<<reg<<endl;
                cout<<"marks="<<m1<<" "<<m2<<" "<<m3<<" "<<m4<<" "<<m5<<endl;
        }

};
class percent:public mark
{
        private:
                float p;
                public:
                        void showpe()
                        {
                                p=(m1+m2+m3+m4+m5)/5;
                                cout<<"percentage="<<p<<"%";
                        }
};
int main()
{
        percent p;
        p.gets();
        p.getm();
        p.shows();
        p.show();
   p.showpe();


}
```

Multiple

```
//multiple inheritance
#include<iostream>
using namespace std;
class person
{
        protected:
                string name, gender;
                int age;
                public:
```

```
		void getp()
		{
			cout<<"enter name, gender, age of student"<<endl;
			cin>>name>>gender>>age;
		}
		void showp()
		{
			cout<<"Name= "<<name<<endl;
			cout<<"gender= "<<gender<<endl;
			cout<<"age= "<<age<<endl;
			}
};
class student
{
	protected:
		string reg, course;
		public:
			void gets()
			{
				cout<<"enter reg no. and course "<<reg<<course<<endl;
				cin>>reg>>course;
			}
		void shows()
		{
			cout<<"reg no.= "<<reg<<endl;
			cout<<"course= "<<course<<endl;
		}
};
class mark: public person, public student
{
	private:
		int m1,m2,m3,m4,m5;
		public:
		void getm()
		{
			cout<<"enter 5 subject marks"<<endl;
			cin>>m1>>m2>>m3>>m4>>m5;
		}
		void showm()
		{
			cout<<"reg= "<<reg<<endl;
			cout<<"marks="<<m1<<" "<<m2<<" "<<m3<<" "<<m4<<" "<<m5<<endl;
		}

};

int main()
{
```

```
        mark p;
        p.getp();
        p.gets();
        p.getm();
        p.showp();
        p.shows();
   p.showm();
        }
```

Hirerarchy inheritance

```
//hirerachy inheritance
#include<iostream>
using namespace std;
class person
{
        protected:
                string name, gender;
                int age;
                public:
                void getp()
                {
                        cout<<"enter name, gender, age of student"<<endl;
                        cin>>name>>gender>>age;
                }
                void showp()
                {
                        cout<<"Name= "<<name<<endl;
                        cout<<"gender= "<<gender<<endl;
                        cout<<"age= "<<age<<endl;
                        }
};
class student:public person
{
        private:
                string clz, reg, course;
                public:
                        void gets()
                        {
                                cout<<"enter clz, reg no. and course "<<clz<<reg<<course<<endl;
                                cin>>clz>>reg>>course;
                        }
                void shows()
                {
                        cout<<"college= "<<clz<<endl;
                        cout<<"reg no.= "<<reg<<endl;
                        cout<<"course= "<<course<<endl;
                }
```

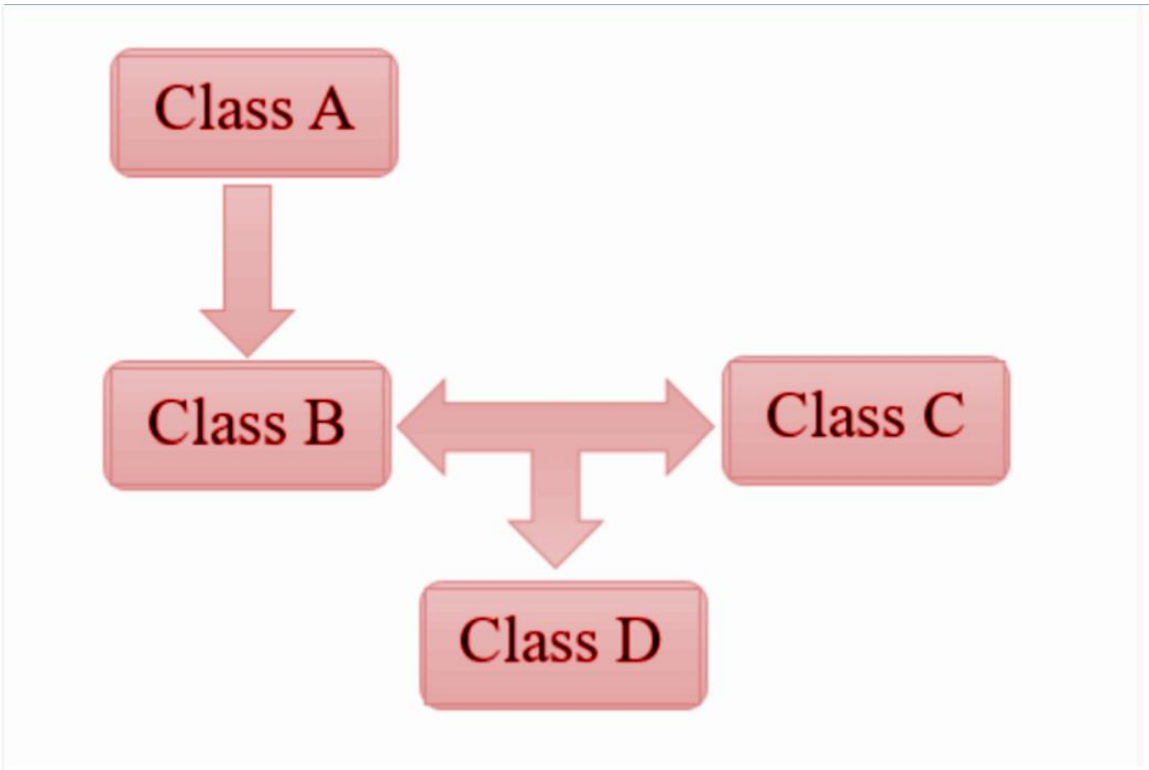
```
};
class employee: public person
{
	private:
		string empid, depart;
		public:
		void gete()
		{
			cout<<"enter empid and department"<<endl;
			cin>>empid>>depart;
		}
		void showe()
		{
			cout<<"employee id = "<<empid<<endl;
			cout<<"department = "<<depart<<endl;
		}

};

int main()
{
	student s;
	cout<<"student detail"<<endl;
	s.getp();
	s.gets();
	s.showp();
	s.shows();
	cout<<"employee detail"<<endl;
	employee E;
	E.getp();
	E.gete();
   E.showp();
   E.showe();

}
```

Hybrid inheritance

```
//hybrid inheritance
#include<iostream>
using namespace std;
class person
{
	protected:
		string name, gender;
		int age;
		public:
		void getp()
```

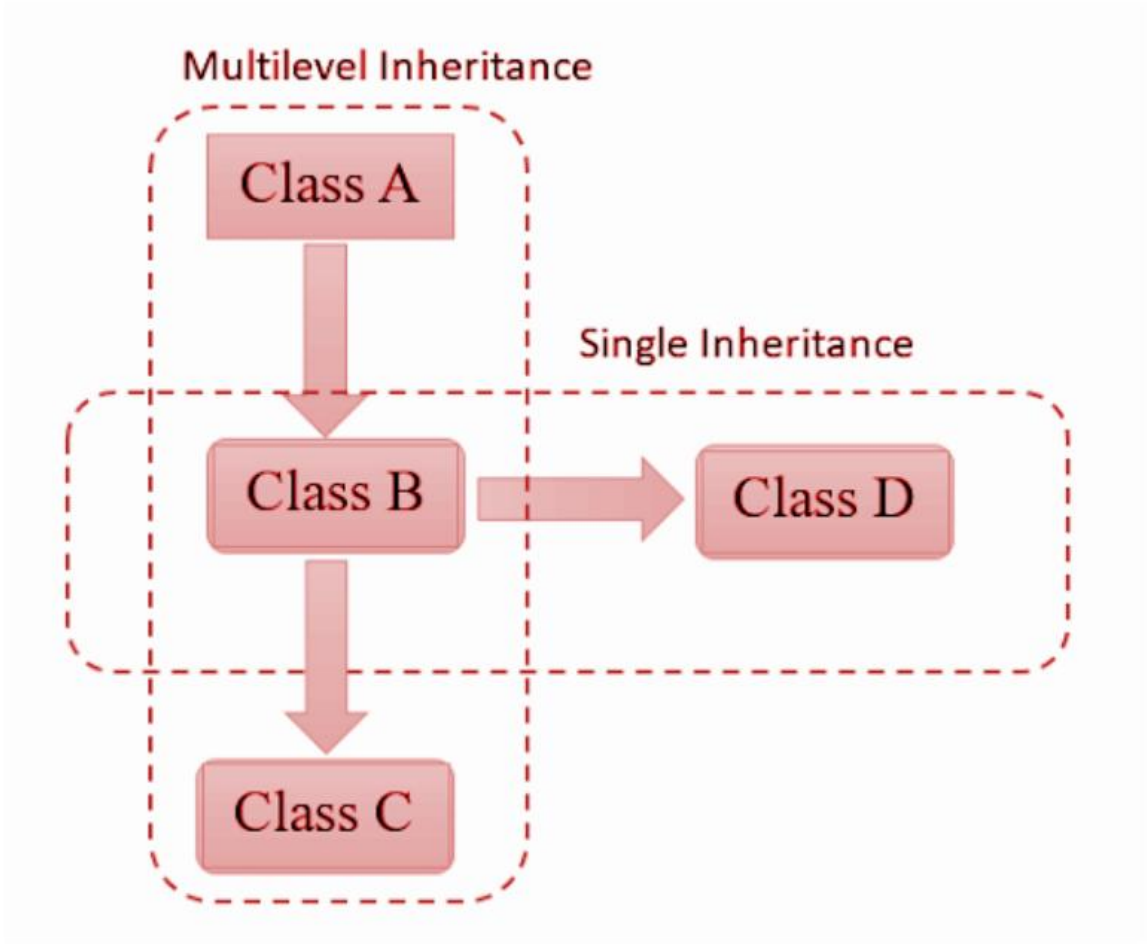
```
		{
			cout<<"enter name, gender, age of student"<<endl;
			cin>>name>>gender>>age;
		}
		void showp()
		{
			cout<<"Name= "<<name<<endl;
			cout<<"gender= "<<gender<<endl;
			cout<<"age= "<<age<<endl;
			}
};
class employee:public person
{
	private:
		string empid, depart;
		public:
		void gete()
		{
			getp();
			cout<<"enter empid and department"<<endl;
			cin>>empid>>depart;
		}
		void showe()
		{
			showp();
			cout<<"employee id = "<<empid<<endl;
			cout<<"department = "<<depart<<endl;
		}

};
class student
{
	protected:
		string clz, reg, course;
		public:
			void gets()
			{
				cout<<"enter clz, reg no. and course "<<clz<<reg<<course<<endl;
				cin>>clz>>reg>>course;
			}
		void shows()
		{
			cout<<"college= "<<clz<<endl;
			cout<<"reg no.= "<<reg<<endl;
			cout<<"course= "<<course<<endl;
		}
};
class marks:public employee, public student
```

```
{
        private:
        int m1,m2,m3,m4,m5;
        public:
        void getm()
        {
                gete();
                gets();
                cout<<"enter 5 subject marks"<<endl;
                cin>>m1>>m2>>m3>>m4>>m5;
        }
        void showm()
        {
                showe();
                shows();
                cout<<"reg= "<<reg<<endl;
                cout<<"marks="<<m1<<" "<<m2<<" "<<m3<<" "<<m4<<" "<<m5<<endl;
        }

};
int main()
{

   marks m;
   m.getm();
   m.showm();
}
```



The diagram shows the hybrid inheritance that is a combination of single inheritance and multiple inheritance.

You can make multilevel and single



This example shows the combination of multilevel and single inheritance.

syntax for above diagram

```
Class A
{
statement(s);
};
Class B: public A
{
statement(s);
};
Class C: public B
{
statement(s);
};
Class D: public B
{
statement(s);
};
```

15) Develop a C++ program to illustrate the order of execution of constructorsand destructors in inheritance.

Code:

```
//order of execution of constructors and destructors in inheritance.
#include<iostream>
using namespace std;
class A
{
        public:
        A()
        {
          cout<<"class a constructor is involved"<<endl;

        }
        ~A()
        {
                cout<<"class a destructor is involved"<<endl;
        }
};
class B:public A
{

   public:
        B()
        {
        cout<<"class B costructor is involved"<<endl;
        }
        ~B()
        {
        cout<<"class B destructor is involved"<<endl;
        }
};
int main()
{
        B b;

}
```

16. Develop a C++ program to illustrate pointer to a class

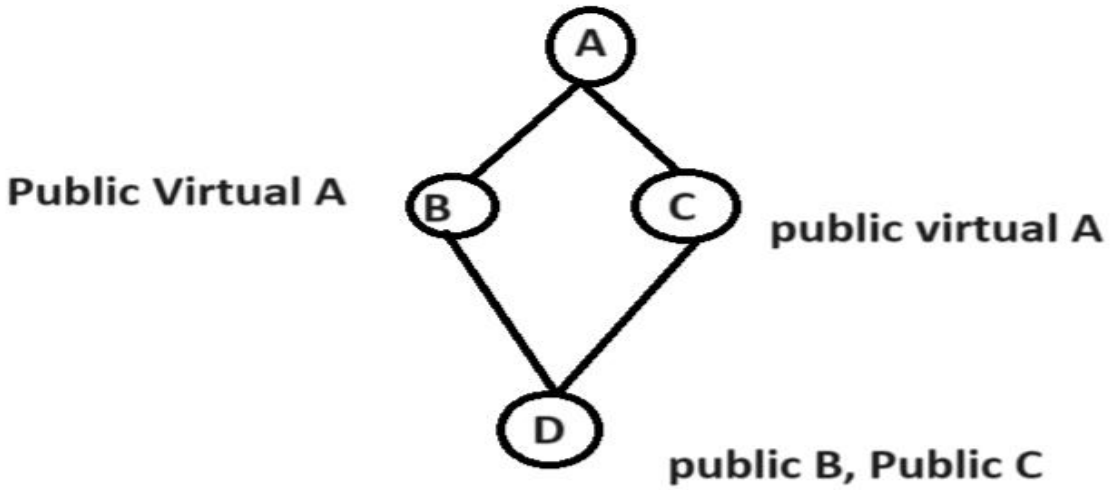
```
#include<iostream>
using namespace std;
class base
{
        public:
         void showb()
        {
                cout<<"base class is involved"<<endl;
```

```
        }
};
class derived: public base
{
        public:
          void showd()
        {
                cout<<"derived class is involved"<<endl;
        }
};
int main()
{
        base b,*bptr;
        derived d,*dptr;
        bptr=&b;
        bptr->showb();
        dptr=&d;
        dptr->showd();
}
```

17. Develop a C++ program to illustrate Virtual Base Class.or multipath or diamond problem.

```
#include<iostream>
using namespace std;
class A
{
        public:
                void showa()
                {
                        cout<<"class A member function is involved"<<endl;
                }
};
class B:public virtual A
{
        public:
                void showb()
                {
                        cout<<"class B member function is involved"<<endl;
                }
};
class C:public virtual A
{
        public:
                void showc()
                {
                        cout<<"class C member function is involved"<<endl;
                }
};
class D:public B, public C
```

```
{
        public:
                void showd()
                {
                        cout<<"class D member function is invlolved"<<endl;
                }
};
int main()
{
        D d;
        d.showa();
        d.showb();
        d.showc();
        d.showd();
}
```



18. Develop a C++ program to illustrate virtual functions.
Or
Develop a C++ program to illustrate runtime polymorphism.

```
#include<iostream>
using namespace std;
class base
{
        public:
                virtual void show()
                {
                        cout<<"Base class member function is involved"<<endl;
                }
```

```
};
class derived:public base
{
	public:
		void show()
		{
			cout<<"Derived class member function is involved"<<endl;
		}
};
int main()
{
	base b, *bptr;
	derived d;
	bptr=&b;
	bptr->show();
	bptr=&d;
	bptr->show();
}
```

19. Develop a C++ program to illustrate pure virtual function and calculate the area of different shapes by using abstract class.(pure virtual function and abstract class program is same)

```
//pure virtual fuction or abstract class
#include<iostream>
using namespace std;
class shape
{
	public:
		virtual void area()=0;
};
class circle:public shape
{
	private:
		float r;
	public:
		void area()
		{
			cout<<"Enter radius of circle: "<<endl;
			cin>>r;
			cout<<"Area of circle is "<<3.14*r*r<<endl;
		}
};
class square:public shape
{
	private:
		int side;
	public:
```

```
		void area()
		{
			cout<<"Enter side of square: "<<endl;
			cin>>side;
			cout<<"Area of circle is "<<side*side<<endl;
		}
};
class rectangle:public shape
{
	private:
		int l,b;
	public:
		void area()
		{
			cout<<"Enter length and breadth of rectangle: "<<endl;
			cin>>l>>b;
			cout<<"Area of circle is "<<l*b<<endl;
		}
};
int main()
{
	shape *sptr;
	circle c;
	sptr=&c;
	sptr->area();
	square s;
	sptr=&s;
	sptr->area();
	rectangle r;
	sptr=&r;
	sptr->area();
}
```

20. Develop a C++ Program illustrating function template

```
//bubble sort using function template

#include<iostream>
using namespace std;
template <class t>
void sort(t a[], int n)
{
	t temp;
	for(int i=0; i<n; i++)
	{
		for(int j=0; j<n-1-i; j++)
		{
			if(a[j]>a[j+1])
			{
```

```
                                temp=a[j];
                                a[j]=a[j+1];
                                a[j+1]=temp;
                        }
                }
        }
        for(int i=0; i<n; i++)
        {
                cout<<a[i]<<endl;
        }
}
int main()
{
        int a[5]={9, 1, 5, 0, 3};
        float f[5]={9.9f, 1.1f, 5.55f, 0.1f, 3.5f};
        cout<<"intger sorted array is: "<<endl;
        sort(a,5);
        cout<<"float sorted array is: "<<endl;
        sort(f,5);
        }
```

21. Develop a C++ Program illustrating template class

```
#include<iostream>
using namespace std;
template <class t>
class sample
{
        private:
                t n;
                public:
                        void get()
                        {
                                cout<<"enter n value: "<<endl;
                                cin>>n;
                        }
                        void show()
                        {
                                cout<<"n value is : "<<n<<endl;
                        }
};
int main()
{
        sample <int> s1;
        s1.get();
        s1.show();
        sample <float> s2;
        s2.get();
        s2.show();
```

```
        sample<char> s3;
        s3.get();
        s3.show();
        sample<string> s4;
        s4.get();
        s4.show();
}
```

22. Develop a C++ program to illustrate class templates with multiple parameters

```
//Develop a C++ program to illustrate class templates with multiple parameters
#include<iostream>
using namespace std;
template <class t1, class t2>
class sample
{
        private:
                t1 a;
                t2 b;
                public:
                        void get()
                        {
                                cout<<"enter a value : "<<endl;
                                cin>>a;
                                cout<<"enter b value : "<<endl;
                                cin>>b;

                        }
                        void show()
                        {
                                cout<<" a value is :" <<a<<endl;
                                cout<<" b value is : " <<b<<endl;
                        }
};
int main()
{
        sample <int , float> s1;
        s1.get();
        s1.show();
        sample<float,int> s2;
        s2.get();
        s2.show();
        sample <int, string> s3;
        s3.get();
```

```
        s3.show();
}
```

23. Develop a C++ program for handling Exceptions

```
#include<iostream>
using namespace std;
int main()
{
        int a,b;
        cout<<"Enter a and b value"<<endl;
        cin>>a>>b;
        int x=a-b;
                try
                {
                        if(x!=o)
                        {
                         cout<<"result: "<<a/x<<endl;
                        }
                        else
                        {
                         throw(x);
                        }

                }
                catch(int i)
                {
                        cout<<"exception catch x= "<<x<<endl;
                }
                cout<<"end";
}
```

24. Develop a C++ program to illustrate the use of multiple catch statements.

```
#include<iostream>
using namespace std;
void test(int x)
{
        try
        {
                if(x!=o)
                {
                        throw(x);
```

```
                }
                else
                {
                        throw ('x');
                }
        }
        catch(int x)
        {
                cout<<"catch a integer and that integer is x : "<<x<<endl;
        }
        catch(char x)
        {
                cout<<"catch a string and that string x : "<<x<<endl;
        }

}
int main()
{
        test(10);
        test(0);
}
```

25. Develop a C++ program to implement List, Vector and its Operations.

Vector

```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
        vector <int> v;
        for(int i=1; i<=5; i++)
        v.push_back(i);
        cout<<"size= "<<v.size()<<endl;
        v.insert(v.begin()+3,50);
        v.erase(v.begin());
        vector <int>::iterator it;
        for(it=v.begin(); it!=v.end(); it++)
        {
                cout<<*it<<" ";
        }
        cout<<"after deletion"<<endl;
        v.pop_back();
        for(it=v.begin(); it!=v.end(); it++)
        {
                cout<<*it<<" ";
        }
```

}

List

```
#include<iostream>
#include<list>
using namespace std;
int main()
{
        list <int> l;
        for(int i=1; i<=5; i++)
        l.push_back(i);
        for(int i=1;i<=5;i++)
        {
        l.push_front(5*i);
        }
        cout<<"size= "<<l.size()<<endl;
        l.insert(l.begin(),50);
        l.erase(l.begin());
        list <int>::iterator it;
        for(it=l.begin(); it!=l.end(); it++)
        {
                cout<<*it<<" ";
        }
        cout<<"after deletion"<<endl;
        l.pop_back();
        l.pop_front();
        for(it=l.begin(); it!=l.end(); it++)
        {
                cout<<*it<<" ";
        }

}
```

26. Develop a C++ program to implement Deque and Deque Operations

DEQUE

```
#include<iostream>
#include<deque>
using namespace std;
int main()
{
```

```
        deque <int> l;
        for(int i=1; i<=5; i++)
        l.push_back(i);
        for(int i=1;i<=5;i++)
        {
        l.push_front(5*i);
        }
        cout<<"size= "<<l.size()<<endl;
        l.insert(l.begin(),50);
        l.erase(l.begin());
        deque <int>::iterator it;
        for(it=l.begin(); it!=l.end(); it++)
        {
                cout<<*it<<" ";
        }
        cout<<"after deletion"<<endl;
        l.pop_back();
        l.pop_front();
        for(it=l.begin(); it!=l.end(); it++)
        {
                cout<<*it<<" ";
        }


}
```

27. Develop a C++ program to implement Map , set Operations.

SET

```
#include<iostream>
#include<set>
using namespace std;
int main()
{
        set <int> v;
        for(int i=1; i<=5; i++)
        v.insert(i);
        cout<<"size="<<v.size()<<endl;
        set <int>::iterator it;
        for(it=v.begin(); it!=v.end(); it++)
        {
                cout<<*it<<" ";
```

```
        }
    }
```

MAP

```
#include<iostream>
#include<map>
using namespace std;
int main()
{
    map <string,int> m;
    m["abc"]=100;
    m["a"]=200;
    m["b"]=300;
    map <string, int>::iterator it;
    for(it=m.begin(); it!=m.end(); it++)
    {
        cout<<it->first<<" "<<it->second<<endl;
    }
}
```