1. **Write a program to print the sum of digits of a given number.**

```
#include<iostream>
using namespace std;
int main()
{
intval, num, sum = 0;
cout<< "Enter the number : ";
cin>>val;
num = val;
while (num != 0)
  {
sum = sum + num % 10;
num = num / 10;
  }
cout<< "The sum of the digits of "
<<val<< " is " << sum;
}
```

OUTPUT:

```
Enter the number : 123
The sum of the digits of 123 is 6
```

2. **Write a program to check whether the given number is Armstrong or not**

```
#include <iostream>
using namespace std;
int main()
{
intorigNum, num, rem, sum = 0;
cout<< "Enter a positive  integer: ";
cin>>origNum;
num = origNum;
while(num != 0)
 {
rem = num % 10;
sum += rem * rem * rem;
num /= 10;
 }
if(sum == origNum)
cout<<origNum<< " is an Armstrong number.";
else
cout<<origNum<< " is not an Armstrong number.";
return 0;
}
```

OUTPUT:

```
Enter a positive integer: 371
371 is an Armstrong number.
```

3. **Write a program to print the prime number from 2 to n where n is natural number given.**

```
#include <iostream>
using namespace std;
int main()
{
int low, high, i, flag;
cout<< "Enter two numbers(intervals): ";
cin>> low >> high;
cout<< "Prime numbers between " << low << " and " << high << " are: ";
while (low < high)
   {
flag = 0;
for(i = 2; i <= low/2; ++i)
      {
if(low % i == 0)
         {
flag = 1;
break;
         }
      }
if (flag == 0)
cout<< low << " ";
      ++low;
   }
return 0;
}
```

OUTPUT:

Enter two numbers(intervals): 2
15
Prime numbers between 2 and 15 are: 2 3 5 7 11 13

4. Write a program to find largest and smallest elements in a given list of numbers and sort the given list.

```
#include<iostream>
using namespace std;
int main ()
{
intarr[10], n,temp, i,j, max, min;
cout<< "Enter the size of the array : ";
cin>> n;
cout<< "Enter the elements of the array : ";
for (i = 0; i < n; i++)
cin>>arr[i];
cout<<"Unsorted Array elements:"<<endl;
         for(i=0;i<n;i++)
                  cout<<arr[i]<<"\t";
         cout<<endl;
max = arr[0];
for (i = 0; i < n; i++)
   {
if (max <arr[i])
max = arr[i];
   }
```

```
min = arr[0];
for (i = 0; i < n; i++)
  {
if (min >arr[i])
min = arr[i];
  }
cout<< "Largest element : " << max<<endl;
cout<< "Smallest element : " << min<<endl;
for(i=0;i<n;i++)
        {
                for(j=i+1;j<n;j++)
                {
                        if(arr[i]>arr[j])
                        {
                                temp  =arr[i];
                                arr[i]=arr[j];
                                arr[j]=temp;
                        }
                }
        }
        cout<<"Sorted Array elements:"<<endl;
        for(i=0;i<n;i++)
                cout<<arr[i]<<"\t";
        cout<<endl;
return 0;
}
```

OUTPUT:

```
Enter the size of the array : 5
Enter the elements of the array : 9
1
3
4
7
Unsorted Array elements:
9       1       3       4       7
Largest element : 9
Smallest element : 1
Sorted Array elements:
1       3       4       7       9
```

5. **Write a program to read the student name, roll no, marks and display the same using class and object.**

```
#include <iostream>
using namespace std;
class student
{
        private:
                char  name[30];
                introllNo;
                int   total;
                floatperc;
        public:
```

```
                voidgetDetails(void);
                voidputDetails(void);
};
void student::getDetails(void)
{
        cout<< "Enter name: " ;
        cin>> name;
        cout<< "Enter roll number: ";
        cin>>rollNo;
        cout<< "Enter marks outof 500: ";
        cin>> total;
}
void student::putDetails(void)
{
        cout<<"Student details:\n";
        cout<<"Name:"<< name <<endl;
        cout<< "Roll Number:" <<rollNo<<endl;
        cout<< "Marks:" << total <<endl;
}
int main()
{
        student std;            //object creation
        std.getDetails();
        std.putDetails();
        return 0;
}
```

OUTPUT:
Enter name: Waseem
Enter roll number: 23
Enter marks outof 500: 470

Student details:
Name:Waseem
Roll Number:23
Marks:470

6. Write a program to implement the dynamic memory allocation and de-allocation using new and delete operators using class and object.

```
#include<iostream>
using namespace std;
classAlloc
{
        public:
                void enter(intx,int y)
                {
                        cout<<"sum="<<x+y;
                }

};
int main()
{
Alloc *p;
```

```
p=new Alloc();
p->enter(5,7);
delete(p);
return 0;
}
```

OUTPUT:

sum=12

7. **Write a program to find area of a rectangle, circle, and square using constructors.**

```
#include<iostream>
#include<math.h>
#include<cstdlib>
using namespace std;
class area
{
    float ar;
    public:
        area(float r)
        {       ar=3.14*r*r;            }
        area(float l, float b)
        {       ar=l*b;             }
        area(float a, float b, float c)
        {
            float s;
            s=(a+b+c)/2;
            ar=s*(s-a)*(s-b)*(s-c);
            ar=pow(ar,0.5);
        }
        void display()
        {
            cout<<"\n Area : "<<ar;
        }
};
int main()
{
    intch;
    float x, y, z;
    do
    {
        <<"\n\n 1. Area of Circle";
        cout<<"\n 2.Area of Rectangle";
        cout<<"\n 3.Area of Triangle";
        cout<<"\n 4. Exit";
        cout<<"\n\n Enter Your Choice : ";
        cin>>ch;
        switch(ch)
        {
            case 1 :
            {
                cout<<"\n Enter Radius of the Circle : ";
                cin>>x;
                area a1(x);  //Class area, object is created : a1
                a1.display();
```

```
                }
                break;
                case 2 :
                {
                    cout<<"\n Enter Length and Breadth of the Rectangle : ";
                    cin>>x>>y;
                    area a2(x,y);
                    a2.display();
                }
                break;
                case 3 :
                {
                    cout<<"\n Enter Sides of the Triangle : ";
                    cin>>x>>y>>z;
                    area a3(x,y,z);
                    a3.display();
                }
                break;
                case 4 :
                    exit(0);

                default :
                    cout<<"\n\n Invalid Choice ..";
            }
        } while(ch!=4);
        return 0;
}
```

OUTPUT:

1. Area of Circle
2. Area of Rectangle
3. Area of Triangle
4. Exit

Enter Your Choice : 1

Enter Radius of the Circle : 5

Area : 78.5

1. Area of Circle
2. Area of Rectangle
3. Area of Triangle
4. Exit

Enter Your Choice :4

8. Write a program to implement copy constructor.

Same as 12th Program

9. Write a program using friend functions and friend class

```
#include<iostream.h>
```

```
class Rectangle
{
        int L,B;
        public:
        Rectangle()
        {
                L=10;
                B=20;
        }
        friend class Square;      //Statement 1
};
class Square
{
        int S;
        public:
        Square()
        {
                S=5;
        }
        void Display(Rectangle Rect)
        {
                cout<<"\n\n\tLength : "<<Rect.L;
                cout<<"\n\n\tBreadth : "<<Rect.B;
                cout<<"\n\n\tSide : "<<S;
        }
};
void main()
{
        Rectangle R;
        Square S;
        S.Display(R);      //Statement 2
}
```

Output :

```
Length : 10
Breadth : 20
Side : 5
```

10. Write a program to implement default Constructor.

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

class Student
{

int Roll;
char Name[25];
float Marks;
public:
Student()        //Default Constructor
{
Roll = 1;
strcpy(Name,"Kumar");
```

```
Marks = 78.42;
}
void Display()
{
        cout<<"\n\tRoll : "<<Roll;
        cout<<"\n\tName : "<<Name;
        cout<<"\n\tMarks : "<<Marks;
}
};
void main()
{
        Student S;      //Creating Object
        S.Display();    //Displaying Student Details
}
```

Output :

```
Roll : 1
Name : Kumar
Marks : 78.42
```

11. Write a program to implement parameterized Constructor

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class Student
{
        int Roll;
        char Name[25];
        float Marks;
        public:
        Student(intr,char nm[],float m)      //Parameterize Constructor
        {
                Roll = r;
                strcpy(Name,nm);
                Marks = m;
        }
void Display()
        {
                cout<<"\n\tRoll : "<<Roll;
                cout<<"\n\tName : "<<Name;
                cout<<"\n\tMarks : "<<Marks;
        }
};
void main()
{
        Student S(2,"Sumit",89.63);
        //Creating Object and passing values to Constructor
        S.Display();
        //Displaying Student Details
}
```

Output :

```
                Roll : 2
                Name :Sumit
                Marks : 89.63
```

12. **Write a program to implement Copy Constructor**

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class Student
{
        int Roll;
        char Name[25];
        float Marks;
        public:
        Student(intr,char nm[],float m)   //Constructor 1 : Parameterize Constructor
        {
                Roll = r;
                strcpy(Name,nm);
                Marks = m;
        }
        Student(Student &S)   //Constructor 2 : Copy Constructor
        {
                Roll = S.Roll;
                strcpy(Name,S.Name);
                Marks = S.Marks;
        }
        void Display()
        {
                cout<<"\n\tRoll : "<<Roll;
                cout<<"\n\tName : "<<Name;
                cout<<"\n\tMarks : "<<Marks;
        }
};
void main()
{

        Student S1(2,"Sumit",89.63);

        Student S2(S1);    //Statement 1
        cout<<"\n\tValues in object S1";
        S1.Display();
        cout<<"\n\tValues in object S2";
        S2.Display();
}
```

OUTPUT :

```
Values in object S1
Roll : 2
Name :Sumit
Marks : 89.63

Values in object S2
Roll : 2
Name :Sumit
Marks : 89.63
```

13. **Write a program to define the constructor inside/outside of the class**

Inside the class	Outside the class

#include <iostream>
usingnamespacestd;
class Example
{
public:
Example()
{
cout<<"Constructor called."<<endl;
}
void display()
{
cout<<"display function called."<<endl;
}
~Example()
{
cout<<"Destructor called."<<endl;
}
};
 	
intmain()
{
Example objE;
objE.display();
return0;
}
OUTPUT:
Constructor called.
display function called.
Destructor called.
	#include <iostream>
	usingnamespacestd;
	class Example
	{
	public:
	Example();
	void display();
	~Example();
	};
	Example::Example()
	{cout<<"Constructor called."<<endl; }
	void Example::display()
	{cout<<"display function called."<<endl; }
	Example::~Example()
	{
	cout<<"Destructor called."<<endl;
	}
	
	intmain()
	{
	Example objE;
	objE.display();
	return0;
	}
	OUTPUT:
	Constructor called.
	display function called.
	Destructor called.

14. Write a program to implement all three constructors within a single class as well as use multiple classes(individual classes)

```
#include<iostream>
using namespace std;
class Distance
{
        private:
        intfeet,inch;
public:
        Distance ()
        {
                feet=0;
                inch=0;
                }
        Distance (inta,int b)
          {
                feet=a;
                inch=b;
                }
        Distance (Distance p,Distance q)
          {
```

```
                feet=p.feet+q.feet;
                inch=p.inch+q.inch;
                if(inch>=12)
                {
                        inch=inch-12;
                        feet++;
                        }
                }
                void show()
                {
                        cout<<feet<<" Feet and ";
                        cout<<inch<<" Inches \n";
                }
};

classDemo
{
        public:
                void print()
                {
                        Distancep(5,8);
                        Distanceq(2,6);
                        Distancer(p,q);
                        p.show();
                        q.show();
                        r.show();
                }

};
int main()
{
Demo m;
m.print();
return 0;
}
```

Output:-

```
5 Feet and 8 Inches
2 Feet and 6 Inches
8 Feet and 2 Inches
```

15. Write a program to implement the following concepts using class and object

a. Function overloading

b. Operator overloading (unary/binary(+ and -))

FUNCTION OVERLOADING

```
#include <iostream>
using namespace std;
void display(int);
void display(float);
void display(int, float);
int main() {
int a = 5;
float b = 5.5;
```

```
display(a);
display(b);
display(a, b);
return 0;
}
void display(intvar) {
cout<< "Integer number: " <<var<<endl;
}
void display(float var) {
cout<< "Float number: " <<var<<endl;
}
void display(int var1, float var2) {
cout<< "Integer number: " << var1;
cout<< " and float number:" << var2;
}
```

OUTPUT:

```
Integer number: 5
Float number: 5.5
Integer number: 5 and float number: 5.5
```

OPERATOR OVERLOADING

```
include<iostream>
usingnamespacestd;
classComplex {
private:
  intreal, imag;
public:
  Complex(intr = 0, inti =0)  {real = r;   imag = i;}
  Complex operator + (Complex const&obj)
        {
    Complex res;
    res.real = real + obj.real;
    res.imag = imag + obj.imag;
    returnres;
  }
  voidprint() { cout<< real << " + i"<<imag<<endl; }
};



intmain()
{
  Complex c1(10, 5), c2(2, 4);
  Complex c3 = c1 + c2; // An example call to "operator+"
  c3.print();
        }
```

Output:

```
        12 + i9
```

16. Write a program to demonstrate single inheritance, multilevel inheritance and multiple inheritances.

a. single inheritance program

```
#include<iostream>
using namespace std;
class One
{
        public:
                void sum(inta,int b)
                {
                        cout<<"Sum="<<a+b<<endl;
                }
};
classTwo:public One
{
        public:
                void sub(inta,int b)
                {
                        cout<<"Sub="<<a-b<<endl;
                }
};
int main()
{
   Two t;
t.sum(4,6);
t.sub(4,6);
}
```

Output:-

```
Sum=10
Sub=-2
```

b. Multilevel inheritance program

```
#include<iostream>
using namespace std;
class One
{
        public:
                void sum(inta,int b)
                {
                        cout<<"Sum="<<a+b<<endl;
                }
};
classTwo:public One
{
        public:
                void sub(inta,int b)
                {
                        cout<<"Sub="<<a-b<<endl;
                }
};
classThree:public Two
{
        public:
                voidmul(inta,int b)
                {
                        cout<<"Mul="<<a*b<<endl;
```

```
        }
};
int main()
{
        Three r;
        r.sum(4,6);
        r.sub(4,6);
        r.mul(4,6);
}
```

Output:

```
Sum=10
Sub=-2
Mul=24
```

c. Multiple inheritance program

```
#include<iostream>
using namespace std;
class One
{
        public:
                void sum(inta,int b)
                {
                        cout<<"Sum="<<a+b<<endl;
                }
};
class Two
{
        public:
                void sub(inta,int b)
                {
                        cout<<"Sub="<<a-b<<endl;
                }
};
classThree:publicOne,public Two
{
        public:
                voidmul(inta,int b)
                {
                        cout<<"Mul="<<a*b<<endl;
                }
};
int main()
{
        Three r;
        r.sum(4,6);
        r.sub(4,6);
        r.mul(4,6);
}
```

Output:

```
Sum=10
Sub=-2
Mul=24
```

17. **Write a program to implement the overloaded constructors in inheritance.**

```
classBase
{
int x;
public:
// default constructor
Base()
{
cout<<"Base default constructor\n";
}
};

classDerived: public Base
{
int y;
public:
Derived()
{
cout<<"Derived default constructor\n";
}
Derived(int i)
{
cout<<"Derived parameterized constructor\n";
}
};
intmain()
{
   Base b;
   Derived d1;
   Derived d2(10);
}
```

OUTPUT:

```
Base default constructor
Base default constructor
Derived default constructor
Base default constructor
Derived parameterized constructor
```

18. Write a program to implement the polymorphism and the following concepts using class and object.

a. Virtual functions

b. Pure virtual functions

Virtual Functions

```
#include <iostream>
usingnamespacestd;
class Base
{
public:
        virtualvoiddisp()
        {
        cout<<"disp function of Base class"<<endl;
        }
};
```

```
class Derived1:public Base
{
  public:
        voiddisp()
        {
        cout<<"disp function of Derived1 class"<<endl;
        }
};
class Derived2:public Base
{
  public:
        voiddisp()
        {
                cout<<"disp function of Derived2 class"<<endl;
        }
};
intmain()
{
        Base *b;
        Derived1 D1;
        Derived2 D2;
        b=&D1;
        b->disp();
        b=&D2;
        b->disp();
        return0;
}
```

PURE VIRTUAL FUNCTIONS

```
#include<iostream>
usingnamespacestd;

classBase
{
  intx;
public:
  virtualvoidfun() = 0;
  intgetX() { returnx; }
};
classDerived: publicBase
{
  inty;
public:
  voidfun() { cout<< "fun() called"; }
};
intmain(void)
{
  Derived d;
  d.fun();
  return0;
}
```

19. Write a program to implement the virtual concepts for following concepts

a. Constructor (not applied)

b. Destructor (applied)

```
#include <iostream>
Using namespace std;
classBase
{
public:
    Base() { }
    virtual
  ~Base() { }
    Virtual void DisplayAction() = 0;
};
 classDerived1 : publicBase
{
public:
   Derived1()
   {
      cout<< "Derived1 created"<<endl;
   }
    ~Derived1()
   {
      cout<< "Derived1 destroyed"<<endl;
   }
    void DisplayAction()
   {
      cout<< "Action from Derived1"<<endl;
   }
};
 classDerived2 : publicBase
{
public:
   Derived2()
   {
      cout<< "Derived2 created"<<endl;
   }
    ~Derived2()
   {
      cout<< "Derived2 destroyed"<<endl;
   }
    voidDisplayAction()
   {
      cout<< "Action from Derived2"<<endl;
   }
};
classUser
{
public:
   User() : pBase(0)
   {
      pBase = newDerived1();
   }
   ~User()
   {
```

```
    if(pBase )
    {
      deletepBase;
      pBase = 0;
    }
  }
  voidAction()
  {
    pBase->DisplayAction();
  }
private:
  Base *pBase;
};
intmain()
{
  User *user = newUser();
  user->Action();
  deleteuser;
}
Derived1 created
Action from Derived1
Derived1 destroyed
```

Without Virtual Destructor	With Virtual Destructor

#include<iostream>
usingnamespacestd;
classbase {
public:
base()
{ cout<<"Constructing base \n"; }
~base()
{ cout<<"Destructing base \n"; }
};
classderived: publicbase {
public:
derived()
{ cout<<"Constructing derived \n"; }
~derived()
{ cout<<"Destructing derived \n"; }
};
intmain(void)
{
derived *d = newderived();
base *b = d;
deleteb;
getchar();
return0;
}
	#include<iostream>
	usingnamespacestd;
	classbase {
	public:
	base()
	{ cout<<"Constructing base \n"; }
	virtual~base()
	{ cout<<"Destructing base \n"; }
	};
	classderived: publicbase {
	public:
	derived()
	{ cout<<"Constructing derived \n"; }
	~derived()
	{ cout<<"Destructing derived \n"; }
	};
	intmain(void)
	{
	derived *d = newderived();
	base *b = d;
	deleteb;
	getchar();
	return0;
	}

20. Write a program to demonstrate static polymorphism using method overloading.

```
#include<iostream>
usingnamespace std;
classAdd{
public:
int sum(int num1,int num2){
return num1+num2;
}
int sum(int num1,int num2,int num3){
return num1+num2+num3;
}
};
int main(){
Addobj;
cout<<"Output: "<<obj.sum(10,20)<<endl;
cout<<"Output: "<<obj.sum(11,22,33);
return0;
}
```

Output: 30
Output: 66

21. Write a program to demonstrate dynamic polymorphism using method overriding and dynamic method dispatch.

```
#include<iostream>
usingnamespace std;
class A {
public:
voiddisp(){
cout<<"Super Class Function"<<endl;
}
};
class B:public A{
public:
voiddisp(){
cout<<"Sub Class Function";
}
};
int main(){
//Parent class object
Aobj;
obj.disp();
//Child class object
  B obj2;
obj2.disp();
return0;
}
```

OUTPUT:
Super class function
Sub class function

22. Write a program to implement the template (generic) concepts

a. Without template class and object

b. With template class and object

```
#include <iostream>
using namespace std;
template<class T>
class Calculator
{
private:
        T num1, num2;
public:
        Calculator(T n1, T n2)
        {
                num1 = n1;
                num2 = n2;
        }
voiddisplayResult()
        {
                cout<< "Numbers are: " << num1 << " and " << num2 << "." <<endl;
                cout<< "Addition is: " << add() <<endl;
                cout<< "Subtraction is: " << subtract() <<endl;
                cout<< "Product is: " << multiply() <<endl;
                cout<< "Division is: " << divide() <<endl;
        }
        T add() { return num1 + num2; }
        T subtract() { return num1 - num2; }
        T multiply() { return num1 * num2; }
        T divide() { return num1 / num2; }
};
int main()
{
        Calculator<int>intCalc(2, 1);
        Calculator<float>floatCalc(2.4, 1.2);
        cout<< "Int results:" <<endl;
        intCalc.displayResult();
        cout<<endl<< "Float results:" <<endl;
        floatCalc.displayResult();
        return 0;
}
```

OUTPUT:

```
Int results:
Numbers are: 2 and 1.
Addition is: 3
Subtraction is: 1
Product is: 2
Division is: 2
```

Float results:
Numbers are: 2.4 and 1.2.
Addition is: 3.6
Subtraction is: 1.2
Product is: 2.88
Division is: 2