

Question 1:- Write a program to print the Fibonacci series with and without recursion.

Code:-

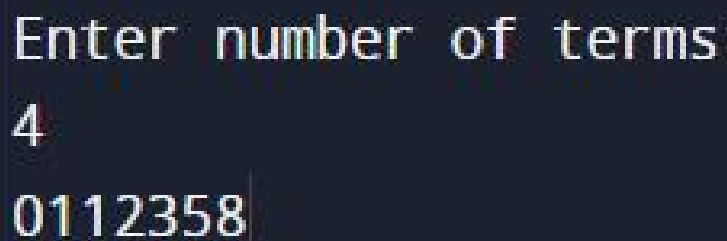
WITHOUT RECURSION:-

```
#include<iostream>

using namespace std;

int main()
{
    int n, temp;
    int t1=0,t2=1;
    int i;
    cout<<"Enter number of terms"<<endl;
    cin>>n;
    cout<<t1<<t2;
    for(i=0; i<=n; i++)
    {
        temp=t1+t2;
        cout<<temp;
        t1=t2;
        t2=temp;
    }
}
```

OUTPUT:-

A screenshot of a terminal window with a dark background. It shows the output of the program: the prompt "Enter number of terms" followed by the user input "4", and then the Fibonacci sequence "0112358" printed on the next line.

```
Enter number of terms
4
0112358
```

WITH RECURSION:-

```
#include<iostream>

using namespace std;

int fib(int);

int main()
{
    int n,i=0;

    cout<<"Enter the number of terms"<<endl;

    cin>>n;

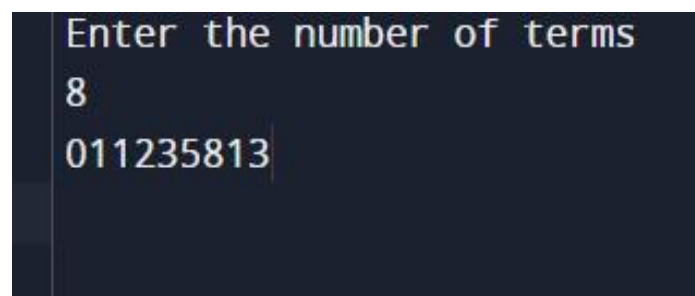
    while(i<n){
        cout<<fib(i);

        i++;
    }
}

int fib(int n)
{
    if(n==1||n==0){
        return n;
    }

    else{
        return(fib(n-1)+fib(n-2));
    }
}
```

OUTPUT:-



```
Enter the number of terms
8
011235813
```

Question 2:- Write a program to print largest and second largest element in an array.

Code:-

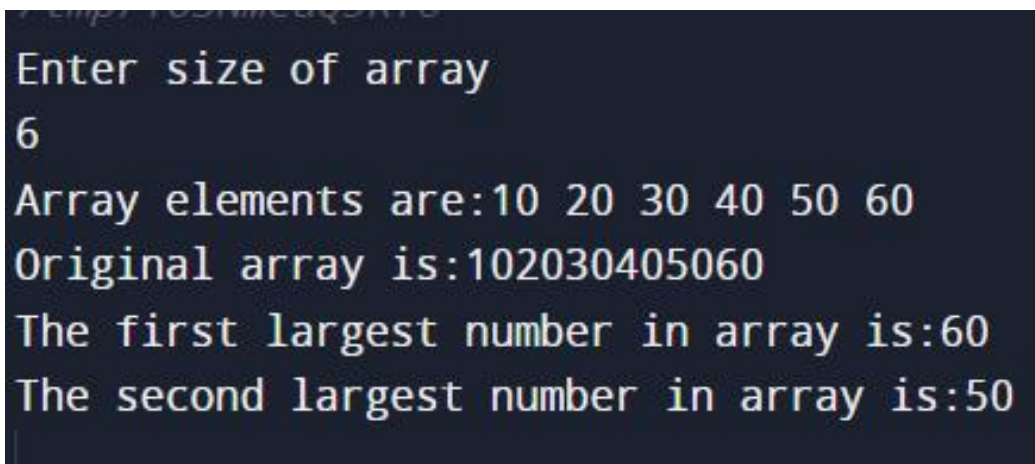
```
#include<iostream>

using namespace std;

int main()
{
    int num[20];
    int size,i,max,max1;
    cout<<"Enter size of array"<<endl;
    cin>>size;
    cout<<"Array elements are:";
    for(i=0; i<size; i++)
    {
        cin>>num[i];
    }
    cout<<"Original array is:";
    for(i=0; i<size; i++)
    {
        cout<<num[i];
    }
    cout<<"\n";
    for(i=0; i<size; i++)
    {
        if(num[i]>max)
        {
            max1 = max;
            max = num[i];
        }
        else if(num[i]>max1 && num[i]<max)
        {
```

```
        max1 = num[i];
    }
}
cout<<"The first largest number in array is:"<<max<<endl;
cout<<"The second largest number in array is:"<<max1<<endl;
return 0;
}
```

OUTPUT:-

A screenshot of a terminal window with a dark background and light-colored text. The output shows the user entering the size of the array as 6, followed by the array elements 10 20 30 40 50 60. It then displays the original array, the first largest number (60), and the second largest number (50).

```
Enter size of array
6
Array elements are:10 20 30 40 50 60
Original array is:102030405060
The first largest number in array is:60
The second largest number in array is:50
```

Question 3:- Write a program to implement call by value , call by reference and call by address.

Code:-

CALL BY VALUE:

```
#include<iostream>

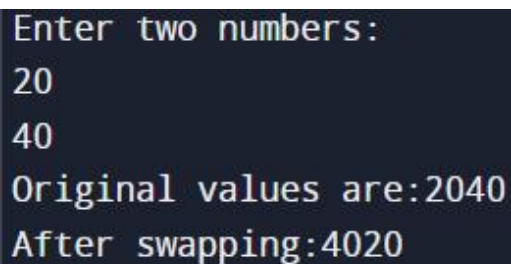
using namespace std;

void swap(int);

int main()
{
    int no1,no2;
    cout<<"Enter two numbers:"<<endl;
    cin>>no1>>no2;
    cout<<"Original values are:"<<no1<<no2<<endl;
    swap(no1,no2);
    cout<<"After swapping:"<<no1<<no2<<endl;
}

int swap(int no1,int no2)
{
    int temp;
    temp = no1;
    no1 = no2;
    no2 = temp;
    cout<<"After swapping :"<<no1<<no2;
}
```

OUTPUT:-

A screenshot of a terminal window with a dark background and light-colored text. The output shows the program's execution: it prompts for two numbers, receives 20 and 40, prints the original values as 20 and 40, and then prints the values after swapping as 40 and 20.

```
Enter two numbers:
20
40
Original values are:2040
After swapping:4020
```

CALL BY ADDRESS:

```
#include<iostream>

using namespace std;

void swap(int *, int *);

int main()
{
    int no1,no2;
    int p1,p2;
    cout<<"Enter two numbers:"<<endl;
    cin>>no1>>no2;
    cout<<"Original values are:"<<no1<<no2<<endl;
    swap(&no1, &no2);
    cout<<"After swapping:"<<no1<<no2<<endl;
}

void swap(int *p1,int *p2)
{
    int temp;
    temp = *p1;
    *p1 = *p2;
    *p2 = temp;
    cout<<"After swapping :"<<*p1<<*p2;
}
```

OUTPUT:-

```
Enter two numbers:
30
50
Original values are:3050
After swapping :5030After swapping:5030
```

CALL BY REFERENCE:

```
#include<iostream>

using namespace std;

void swap(int&, int&);

int main()
{
    int no1,no2;
    int p1,p2;
    cout<<"Enter two numbers:"<<endl;
    cin>>no1>>no2;
    cout<<"Original values are:"<<no1<<no2<<endl;
    swap(no1, no2);
    cout<<"After swapping:"<<no1<<no2<<endl;
}

void swap(int &a,int &b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
    cout<<"After swapping"<<a<<b;
}
```

OUTPUT:-

```
Enter two numbers:
50
60
Original values are:5060
After swapping6050After swapping:6050
```

Question 4:- Write a program to implement default arguments.

Code:-

```
#include<iostream>

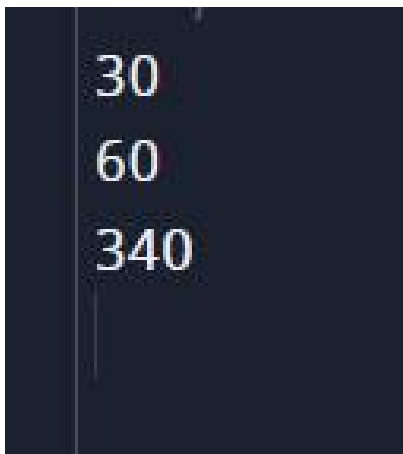
using namespace std;

int sum(int x,int y,int z=0,int w=0)
{
    return(x+y+z+w);
}

int main()
{

    cout<<sum(10,20)<<endl;
    cout<<sum(10,20,30)<<endl;
    cout<<sum(70,80,90,100)<<endl;
}
```

OUTPUT:-

A screenshot of a terminal window with a dark background. It displays the output of the C++ program: the first line shows '30', the second line shows '60', and the third line shows '340'. Each number is on a new line, corresponding to the three cout statements in the code.

```
30
60
340
```


Question 5:- write a program to find area of square, rectangle, sphere and triangle using function overloading.

Code:-

```
#include<iostream>

using namespace std;

void area(int);
void area(int,int);
void area(double);
void area(int,double);

int main()
{
    int side,l,b,base;
    double r,height;
    cout<<"enter the side of the square";
    cin>>side;
    cout<<"enter the lenght and breadth of the rectangle";
    cin>>l>>b;
    cout<<"enter the radius of the sphere";
    cin>>r;
    cout<<"enter the base and height of the triangle";
    cin>>base>>height;
    area(side);
    area(l,b);
    area(r);
    area(base,height);
}

void area(int s)
{
    cout<<s*s<<endl;
}

void area(int l,int b)
```

```

{
    cout<<l*b<<endl;
}
void area(double r)
{
    cout<<4/3*3.14*r*r<<endl;
}
void area(int b,double h)
{
    cout<<1/2*b*h;
}

```

Output:-

```

enter the side of the square4
enter the lenght and breadth of the rectangle4
5
enter the radius of the sphere3.4
enter the base and height of the triangle4
3.5
16
20
36.2984
0

```

Question 6:- WAP to create class employee to store the information as employee id, name, designation, experience, salary by initialising and displaying them using member function. And also find the highest paid employee.

Code:-

```
#include <iostream>
#include <string.h>
using namespace std;

class Employee
{
public:
    int eno;
    char ename[100];
    char des[100];
    int expe;
    float salary;

    void accept_details()
    {
        cout << "Enter Employee Id : ";
        cin >> eno;
        cout << "Enter Employee Name : ";
        cin >> ename;
        cout<<"Enter the designation : ";
        cin>>des;
        cout<<"Enter the experience : ";
        cin>>expe;
        cout << "Enter Salary : ";
        cin >> salary;
    }
}
```

```
};  
  
int main()  
{  
    Employee emp[2];  
    int i, max;  
    for (i = 0; i < 2; i++)  
    {  
        emp[i].accept_details();  
    }  
    max = emp[i].salary;  
    for (i = 0; i < 2; i++)  
    {  
        if (emp[i].salary > max)  
        {  
            max = emp[i].salary;  
        }  
    }  
    for (i = 0; i < 2; i++)  
    {  
        if (emp[i].salary == max)  
        {  
            cout << "\n Maximum Salary of Employee Name is : " << emp[i].ename;  
            cout << "\n And Salary is : " << emp[i].salary;  
        }  
    }  
    return 0;  
}
```

Output:-

```
Enter Employee Id : 1
Enter Employee Name : priyam
Enter the designation : ceo
Enter the experience : 3
Enter Salary : 40000
Enter Employee Id : 2
Enter Employee Name : manjeet
Enter the designation : hr
Enter the experience : 5
Enter Salary : 35000
Maximum Salary of Employee Name is : priyam
And Salary is : 40000
```

Question 7:- WAP to create class student and store the information name, marks, roll_no using private access modifier. Intialise and display the student information using member function by using the concept of object of arrays

Code:-

```
#include<iostream>

using namespace std;

class Student
{
    private:
        int roll_no;
        char name[40];
        float marks;

    public:
        void get()
        {
            cout<<"Enter the Roll_No:";
            cin>>roll_no;
            cout<<"Enter the Name :";
            cin>>name;
            cout<<"Enter the Marks:";
            cin>>marks;
        }
        void display()
        {
            cout<<"Roll_No is:"<<roll_no<<endl;
            cout<<"Name is:"<<name<<endl;
            cout<<"Marks is:"<<marks<<endl;
        }
};

int main()
```

```
{  
    int i;  
    Student S[2];  
    for(i=0;i<2;i++)  
    {  
        S[i].get();  
        S[i].display();  
    }  
}
```

Output:-

```
Enter the Roll_No:1  
Enter the Name :priyam  
Enter the Marks:99  
Roll_No is:1  
Name is:priyam  
Marks is:99  
Enter the Roll_No:|
```

Question 8:- WAP to create a class employee with data members id, name, salary. Create 3 object e1, e2, e3. Initialize e1 with default constructor, e2 with parametrized constructor and e3 with copy constructor and display them all .

Code:-

```
#include <iostream>
#include <string.h>
using namespace std;

class Employee
{
public:
    int id;
    char name[100];
    float salary;

    Employee()
    {
        cout << "Enter Employee Id : ";
        cin >> id;
        cout << "Enter Employee Name : ";
        cin >> name;
        cout << "Enter Salary : ";
        cin >> salary;
    }
    Employee(int i,char n[100],float s)
    {
        id=i;
        strcpy(name,n);
        salary=s;
    }
    Employee(Employee &e)
    {
        id=e.id;
        strcpy(name,e.name);
        salary=e.salary;
    }
    void display()
    {
        cout << "Employee Id : "<<id<<endl;
        cout << "Employee Name : "<<name<<endl;
        cout << "Salary : "<<salary<<endl;
    }
};

int main()
{
```



```
Employee e1;  
e1.display();  
Employee e2(2,"manjeet",15000);  
e2.display();  
Employee e3(e1);  
e3.display();  
}
```

Output:-

```
Enter Employee Id : 1  
Enter Employee Name : Priyam  
Enter Salary : 40000  
Employee Id : 1  
Employee Name : Priyam  
Salary : 40000  
Employee Id : 2  
Employee Name : manjeet  
Salary : 15000  
Employee Id : 1  
Employee Name : Priyam  
Salary : 40000
```

Question 9:-WAP to create class to count no. of objects created.

Code:-

```
#include <iostream>
using namespace std;
class count_obj
{
    static int count;

public:
    count_obj()
    {
        cout << "the number of object created is  " << getcount() << endl;
    }
    static int getcount()
    {
        count = count + 1;
        return count;
    }
    static int getcount1()
    {
        count = count - 1;
        return count;
        // count=count-1;
    }
    ~count_obj()
    {
        cout << "the object is destroyed  " << getcount1() << endl;
    }
};
int count_obj::count = 0;
int main()
{
    count_obj ob1, ob2, ob3;
    return 0;
}
```

Output:-

```
the number of object created is  1
the number of object created is  2
the number of object created is  3
the object is destroyed  2
the object is destroyed  1
the object is destroyed  0
```

Question 10:- WAP to create a dynamic array and calculate the sum of the numbers stored in the array.

Code:-

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

class DynamicArray
{
    int *arr;
    int size;

public:
    DynamicArray(int initialSize)
    {
        size = initialSize;
        arr = new int[size];
    }
    void readNumbers()
    {
        cout << "Enter " << size << " numbers:\n";
        for (int i = 0; i < size; ++i)
        {
            cout << "Enter number " << i + 1 << ": ";
            cin >> arr[i];
        }
    }
    int calculateSum()
    {
        int sum = 0;
        for (int i = 0; i < size; ++i)
        {
            sum += arr[i];
        }
        return sum;
    }
    ~DynamicArray()
    {
        delete[] arr;
    }
};

int main()
{
    int arraySize;
    cout << "Enter the size of the dynamic array: ";
    cin >> arraySize;
```

```
DynamicArray dynamicArray(arraySize);
dynamicArray.readNumbers();
int sum = dynamicArray.calculateSum();
cout << "Sum of the numbers in the array: " << sum << "\n";

return 0;
}
```

Output:-

```
Enter the size of the dynamic array: 2
Enter 2 numbers:
Enter number 1: 10
Enter number 2: 20
Sum of the numbers in the array: 30
```

Question 11:- WAP to compare 2 objects of 2 classes using the concept of friend member function.

Code:-

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

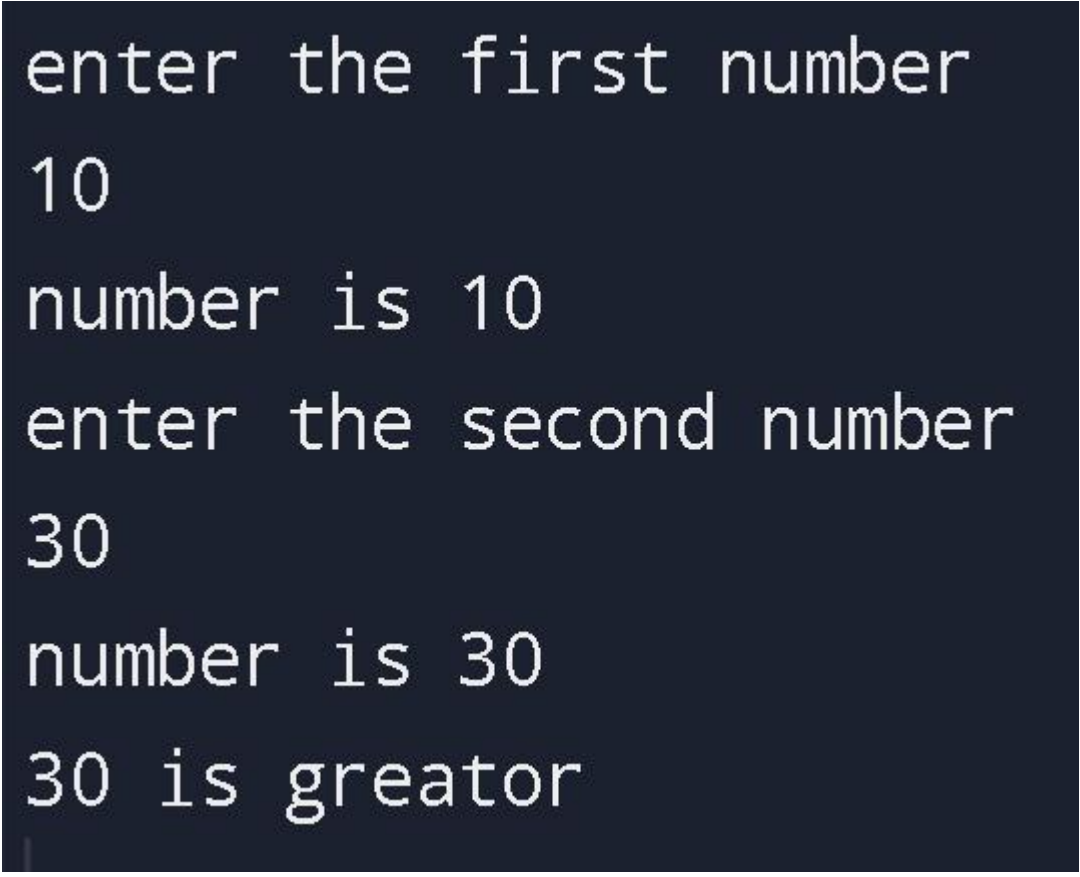
```
class b;
class a
{
    int x;
    public:
    a()
    {
        cout<<"enter the first number"<<endl;
        cin>>x;
        cout<<"number is "<<x<<endl;
    }
    friend void greator(a,b);
};

class b
{
    int y;
    public:
    b()
    {
        cout<<"enter the second number"<<endl;
        cin>>y;
        cout<<"number is "<<y<<endl;
    }
    friend void greator(a,b);
};

void greator(a o1,b o2)
{
    if(o1.x>o2.y)
    {
        cout<<o1.x<<" is greator"<<endl;
    }
    else
    {
        cout<<o2.y<<" is greator"<<endl;
    }
}
```

```
int main()
{
    a ob1;
    b ob2;
    greator(ob1,ob2);
}
```

Output:-

A screenshot of a terminal window with a dark background and light-colored text. The output of the program is displayed as follows:

```
enter the first number
10
number is 10
enter the second number
30
number is 30
30 is greator
```

Question 12:- WAP to implement friend class.

Code:-

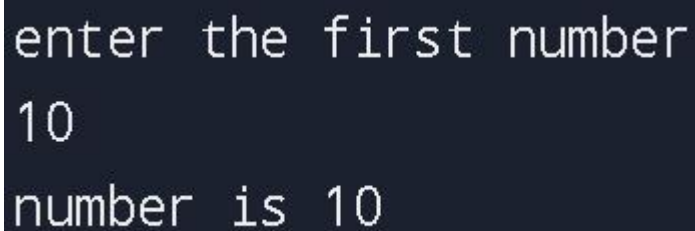
```
#include <iostream>
using namespace std;

class b;
class a
{
    int x;
    public:
    a()
    {
        cout<<"enter the first number"<<endl;
        cin>>x;
    }
    friend class b;
};

class b
{
    public:
    void display(a o)
    {
        cout<<"number is "<<o.x<<endl;
    }
};

int main()
{
    a ob1;
    b ob2;
    ob2.display(ob1);
}
```

Output:-

A screenshot of a terminal window with a dark background and light-colored text. It shows the output of the C++ program: the prompt "enter the first number", the user input "10", and the program's response "number is 10".

```
enter the first number
10
number is 10
```

Question 13:- WAP to create a class complex with 2 data members real and imaginary. Add and Subtract the complex numbers by using operator overloading.

Code:-

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

class complex
{
    int real;
    int imaginary;
public:
    void get()
    {
        cout<<"enter the x number"<<endl;
        cin>>real;
        cout<<"enter the y number"<<endl;
        cin>>imaginary;
    }
    void display()
    {
        cout<<"x is: "<<real<<endl;
        cout<<"y is: "<<imaginary<<endl;
    }
    complex operator+ (complex o)
    {
        complex temp;
        temp.real=real+o.real;
        temp.imaginary=imaginary+o.imaginary;
        return temp;
    }
    complex operator- (complex o)
    {
        complex temp;
        temp.real=real-o.real;
        temp.imaginary=imaginary-o.imaginary;
        return temp;
    }
};

int main()
{
    complex ob1;
    complex ob2;
    complex ob3;
    complex ob4;
```



```
    ob1.get();  
    ob2.get();  
    ob3=ob1+ob2;  
    ob3.display();  
    ob4=ob1-ob2;  
    ob4.display();  
}
```

Output:-

```
enter the x number  
10  
enter the y number  
20  
enter the x number  
30  
enter the y number  
40  
x is: 40  
y is: 60  
x is: -20  
y is: -20
```

Question 14:- WAP to implement:

- **Single inheritance**
- **Multilevel inheritance**
- **Multiple inheritance**
- **Hierarchical inheritance**
- **Hybrid inheritance**

Code:-

• SINGLE INHERITANCE:-

```
#include<iostream>

using namespace std;

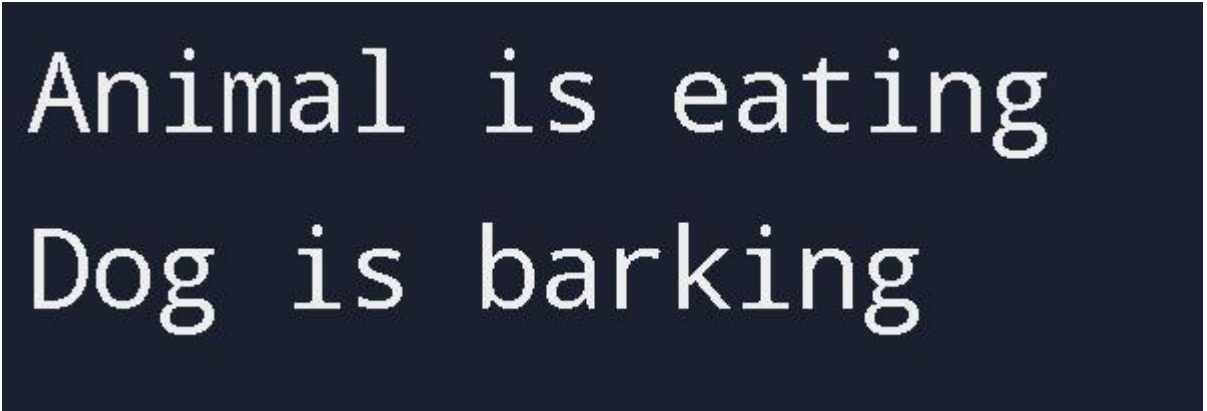
class Animal
{
    char color[10];
public:
    void eat()
    {
        cout<<"Animal is eating"<<endl;
    }
};

class Dog:public Animal
{
    int age;
public:
    void bark()
    {
        cout<<"Dog is barking"<<endl;
    }
};

int main()
{
    Dog ob;
```

```
ob.eat();  
ob.bark();  
  
return 0;  
}
```

Output:-



```
Animal is eating  
Dog is barking
```

Code:-

• **MULTI LEVEL INHERITANCE :-**

```
#include<iostream>  
  
using namespace std;  
  
class Animal  
{  
    char color[10];  
    public:  
    void eat()  
    {  
        cout<<"Animal is eating"<<endl;  
    }  
    void getcolor()  
    {
```

```

        cout<<"Enter color:"<<endl;
        cin>>color;
    }
};
class Dog:public Animal
{
    int age;
public:
    void getage()
    {
        cout<<"Enter age:"<<endl;
        cin>>age;
    }
    void bark()
    {
        cout<<"Dog is barking"<<endl;
    }
};
class Puppy : public Dog
{
    char name[20];
public:
    void getname()
    {
        cout<<"Enter name:"<<endl;
        cin>>name;
    }
    void cry()
    {
        cout<<"Dog is crying";
    }
};

```

```
    }  
};  
int main()  
{  
    Puppy ob;  
    ob.getcolor();  
    ob.eat();  
    ob.getage();  
    ob.bark();  
    ob.getname();  
    ob.cry();  
    return 0;  
}
```

Output:-

```
Enter color:  
black  
Animal is eating  
Enter age:  
12  
Dog is barking  
Enter name:  
hima  
Dog is crying|
```

Code:-**• MULTIPLE INHERITANCE:-**

```
#include<iostream>

using namespace std;

class Animal
{
    char color[10];
public:
    void eat()
    {
        cout<<"Animal is eating"<<endl;
    }
    void getcolor()
    {
        cout<<"Enter color:"<<endl;
        cin>>color;
    }
};

class Bird
{
    int age;
public:
    void getage()
    {
        cout<<"Enter age:"<<endl;
        cin>>age;
    }
    void sing()
    {
        cout<<"Bird is singing"<<endl;
```

```

    }
};
class ABC: public Animal, public Bird
{
    char name[20];
public:
    void getname()
    {
        cout<<"Enter name:"<<endl;
        cin>>name;
    }
    void play()
    {
        cout<<"ABC is playing";
    }
};
int main()
{
    ABC ob;
    ob.getcolor();
    ob.eat();
    ob.getage();
    ob.sing();
    ob.getname();
    ob.play();

    return 0;
}

```

Output:-

```
Enter color:
yello
Animal is eating
Enter age:
11
Bird is singing
Enter name:
pea
ABC is playing
```

Code:-

• **HIERARCHIAL INHERITANCE:-**

```
#include<iostream>
using namespace std;
class shape
{
    public:
    int x,y;
    public:
    void get()
    {
        cout<<"Enter x:";
```



```

        cin>>x;

        cout<<"Enter y:";

        cin>>y;
    }
    void display()
    {
        cout<<"x is:"<<x<<endl;
        cout<<"y is:"<<y<<endl;
    }
};

class rectangle : public shape
{
    int area;

    public:
    int calculate_area()
    {
        area=x*y;
        return area;
    }
};

class triangle : public shape
{
    float a;

    public:
    float area_calc()
    {
        a=0.5*x*y;
        return a;
    }
};

```

```
int main()
{
    float ans1;
    int ans;
    rectangle ob;
    ob.get();
    ob.display();
    ans=ob.calculate_area();
    cout<<"Area of rectangle is:"<<ans<<endl;

    triangle ob1;
    ob1.get();
    ob1.display();
    ans1=ob1.area_calc();
    cout<<"Area of triangle is:"<<ans1;

    return 0;
}
```

Output:-

```
Enter x:10
Enter y:20
x is:10
y is:20
Area of rectangle is:200
Enter x:30
Enter y:40
x is:30
y is:40
Area of triangle is:600
```

Code:-

• HYBRID INHERITANCE:-

```
#include<iostream>

using namespace std;

class Vehicle
{
    int num;
    char m[20];
public:
    void get()
    {
        cout<<"Enter car number:"<<endl;
        cin>>num;
        cout<<"Enter model name:"<<endl;
        cin>>m;
    }
    void display()
    {
        cout<<"Car Number is:"<<num<<endl;
        cout<<"Model name is:"<<m<<endl;
    }
};

class Car : public Vehicle
{
    char n[20];
    char color[40];
public:
    void getcolor()
    {
        cout<<"Enter car color:"<<endl;
```

```

        cin>>color;
    }
    void input()
    {
        cout<<"Enter car name:"<<endl;
        cin>>n;
    }
};

class SportsCar
{
    int speed;
public:
    void getspeed()
    {
        cout<<"Enter car speed:"<<endl;
        cin>>speed;
    }
};

class BMW : public Car,public SportsCar
{
    int seat;
public:
    void getseat()
    {
        cout<<"Enter number of seats:"<<endl;
        cin>>seat;
    }
};

int main()
{

```

```
BMW ob;  
ob.getcolor();  
ob.input();  
ob.get();  
ob.display();  
  
ob.getspeed();  
ob.getseat();  
}
```

Output:-

```
Enter car color:  
black  
Enter car name:  
gt  
Enter car number:  
0001  
Enter model name:  
suv  
Car Number is:1  
Model name is:suv  
Enter car speed:  
200  
Enter number of seats:  
5
```

Question 15:- WAP to implement virtual base class.

Code:-

```
#include <iostream>

using namespace std;

class A
{
    int a;
public:
    A()
    {
        a=10;
    }
    void disp()
    {
        cout<<"a is: "<<a<<endl;
    }
};

class B1: public virtual A
{
    int b;
public:
    B1()
    {
        b=10;
    }
    void disp1()
    {
        cout<<"b is: "<<b<<endl;
    }
}
```

```

};

class B2: public virtual A
{
    int c;
public:
    B2()
    {
        c=20;
    }
    void disp2()
    {
        cout<<"c is: "<<c<<endl;
    }
};

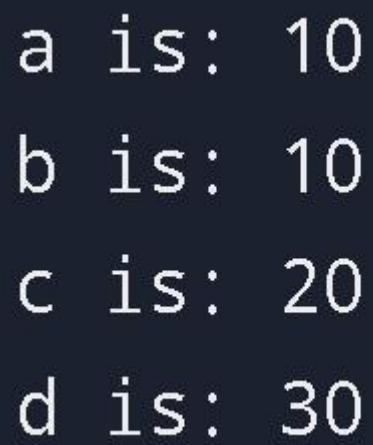
class C: public B1, public B2
{
    int d;
public:
    C()
    {
        d=30;
    }
    void disp3()
    {
        cout<<"d is: "<<d<<endl;
    }
};

int main() {
    C ob;

```

```
    ob.disp();  
    ob.disp1();  
    ob.disp2();  
    ob.disp3();  
    return 0;  
}
```

Output:-

A screenshot of a terminal window with a dark background and light-colored text. It displays four lines of output: 'a is: 10', 'b is: 10', 'c is: 20', and 'd is: 30'. A vertical cursor is visible on the left side of the terminal.

```
a is: 10  
b is: 10  
c is: 20  
d is: 30
```


Question 16:- WAP to create a abstract class shape having data members and member function get() and member function area() (area() is a pure virtual function). Inherit rectangle and triangle from shape and calculate the area

Code:-

```
#include<iostream>
using namespace std;
class shape
{
    protected:
    int a;
    int b;
    public:
    void get()
    {
        cout<<"Enter a:";
        cin>>a;
        cout<<"Enter b:";
        cin>>b;
    }
    virtual void area()=0;
};
class Rectangle : public shape
{
    public:
    void area()
    {
        cout<<"Area of rectangle is:"<<a*b<<endl;
    }
};
class Triangle : public shape
{
    public:
    void area()
    {
        cout<<"Area of triangle is:"<<0.5*a*b;
    }
};
int main()
{
    Rectangle ob1;
    ob1.get();
    ob1.area();

    Triangle ob2;
    ob2.get();
    ob2.area();
}
```

```
    return 0;  
}
```

Output:-

```
Enter a:30  
Enter b:4  
Area of rectangle is:120  
Enter a:45  
Enter b:6  
Area of triangle is:135
```

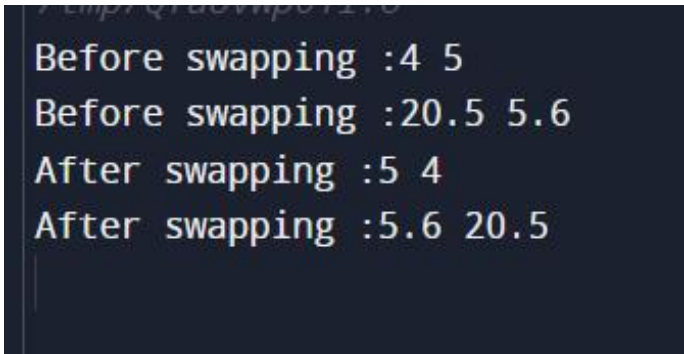
Question 17:- WAP to swap 2 integers and 2 float by creating swap() as a function template

Code:-

```
#include<iostream>
using namespace std;
template<class t>
void swapv (t &a,t &b)
{
    t temp;
    temp=a;
    a=b;
    b=temp;
}
int main()
{
    int a=4,b=5;
    float x=20.5,y=5.6;
    cout<<"Before swapping :"<<a<<" "<<b<<endl;
    cout<<"Before swapping :"<<x<<" "<<y<<endl;
    swapv(a,b);
    cout<<"After swapping :"<<a<<" "<<b<<endl;
    swapv(x,y);
    cout<<"After swapping :"<<x<<" "<<y<<endl;

    return 0;
}
```

Output:-



```
Before swapping :4 5
Before swapping :20.5 5.6
After swapping :5 4
After swapping :5.6 20.5
```

Question 18:- WAP to overload function template

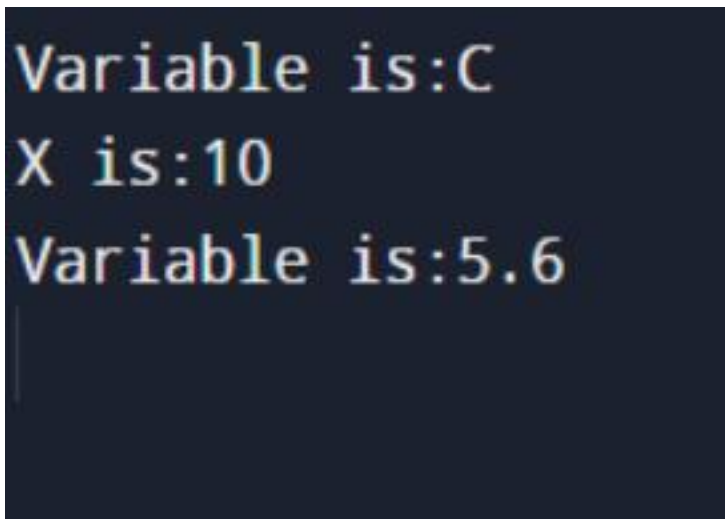
Code:-

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

template<class t>
void display(t a)
{
    cout<<"Variable is:"<<a<<endl;
}
void display(int x)
{
    cout<<"X is:"<<x<<endl;
}
int main()
{
    display('C');
    display(10);
    display(5.6);

    return 0;
}
```

Output:-

A screenshot of a terminal window with a dark background and light-colored text. The output of the program is displayed on three lines: "Variable is:C", "X is:10", and "Variable is:5.6".

```
Variable is:C
X is:10
Variable is:5.6
```

INDEX

S.no.	Program list	signature
1	WAP using c++ to print fibonacci series with and without recursion	
2	WAP using c++ to find the largest and second largest element in an array	
3	WAP to implement call by value, call by address ,call by reference	
4	WAP to implement default arguments in c++	
5	WAP to find area of square, rectangle, sphere and triangle using function overloading	
6	WAP to create class employee to store the information as employee id, name, designation, experience, salary by initialising and displaying them using member function. And also find the highest paid employee.	
7	WAP to create class student and store the information name, marks, roll_no using private access modifier. Intialise and display the student information using member function by using the concept of object of arrays	
8	WAP to create a class employee with data members id, name, salary. Create 3 object e1, e2, e3. Intialize e1 with default constructor, e2 with parametrized constructor and e3 with copy constructor and display them all .	
9	WAP to create class to count no. of objects created.	
10	WAP to create a dynamic array and calculate the sum of the numbers stored in the array.	
11	WAP to compare 2 objects of 2 classes using the concept of friend member function	
12	WAP to implement friend class	
13	WAP to create a class complex with 2 data members real and imaginary. Add and Subtract the complex numbers by using operator overloading	

14	WAP to implement: <ul style="list-style-type: none"> ○ Single inheritance ○ Multilevel inheritance ○ Multiple inheritance ○ Hierarchical inheritance ○ Hybrid inheritance 	
15	WAP to implement virtual base class.	
16	WAP to create a abstract class shape having datamembers and member function get() and member function area() (area() is a pure virtual function). Inherit rectangle and triangle from shape and calculate the area	
17	WAP to swap 2 integers and 2 float by creating swap() as a function template	
18	WAP to overload function template	