C++

1. Write a C++ program to find the area of the circle.

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

#define pi 3.14

int main()
{
    float r,a;
    cout << "Enter radius:";
    cin >> r;
    a = pi*r*r;
    cout << "Area of circle is: " << a;
    return 0;
}
```

2. Write a C++ program to find the area of the rectangle.

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

int main()
{
    float l,b,a;
    cout << "Enter length & base:";
    cin >> l >> b;
    a = l*b;
    cout << "Area of rectangle is: " << a;
    return 0;
}
```

3. Write a C++ program to find the maximum number out of the given 3 numbers.

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

int main()
{
    int a,b,c,max;
    cout << "Enter values:" << endl;
    cin >> a >> b >> c;

    //ladder if else........

     if((a>b) && (a>c)){
        max = a;
    }
    else if((b>c) && (b>a)){
        max = b;
    }
    else{
        max = c;
    }

    cout<<"Max value is: "<<max;
    return 0;
}
```

```
//nested if else........

    if(a>b){
        if(a>c){
            max = a;
        }
        else{
            max = c;
        }
    }
    else{
        if(b>c){
            max = b;
        }
        else{
            max = c;
        }
    }
```

4. Write a C++ program to find a max number from 2 numbers (with teranary operator).

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

int main()
{
    int a,b,max;
    cout << "Enter values:" << endl;
    cin >> a >> b;

    if(a>b){
        max = a;
    }
    else{
        max = b;
    }
```

$\longrightarrow$

```
max = a>b ? a:b;    // teranary op
```

```
    cout << "Max value is: " << max;
    return 0;
}
```

5. Introduction to function :-- Write a C++ program to find the sum of 2 numbers.

```
#include <iostream>
using namespace std;
int add (int x, int y);

int main()
{
    int a,b,ans;
    cout << "Enter 2 values:";
    cin >> a >> b;
    ans = add (a,b);
    cout << "answer is : " << ans << endl;
    return 0;
}

int add (int x, int y){
    int k;
    k = x + y;
    return k;
}
```

6. Write a C++ program to find average of 3 numbers.

```
#include <iostream>
using namespace std;
float avg (int x, int y, int z);

int main()
{
    float ans;
    int a,b,c;
    cout << "Enter 3 values:";
    cin >> a >> b >> c;
    ans = avg (a,b,c);
    cout << "answer is : "<< ans << endl;
    return 0;
}

float avg (int x, int y, int z){
    float k;
    // k = (x + y + z)/3.0;      // implicit type conversion
    k = (float(x + y + z))/3;    // explicit type conversion
    return k;
}
```

7. Funtion with default value/argument :--
   Write a C++ program to find the sum of 2 numbers.

```
#include <iostream>
using namespace std;
int add (int x, int y = 5);     //function with default value

int main()
{
    int ans;
    ans = add(6);
    cout<<"answer is : "<<ans<<endl;
    ans = add(2,10);
    cout<<"answer is : "<<ans<<endl;
    return 0;
}

int add (int x, int y){
    int k;
    k = x + y;
    return k;
}
```

8. Write a C++ program to find area of circle using function(default arguments).

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

float area (int x, float pi = 3.14);     // variable declared as argument
int main()
{
    int x;
    float ans;
    ans = area(2);
    cout << "answer is : " << ans << endl;
    return 0;
}

float area (int x, float pi){
    float k;
    k = float(pi*x*x);
    return k;
}
```

9. Introduction to for loop :-- Write a C++ program to generate table of given number.

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

int main()
{
    int i,k,a;
    cout << "table of : ";
    cin >> a;

    for (i = 1; i <= 10; i++)
    {
        k = a * i;
        cout << a << " x " << i << " = " << k << endl;
    }
    return 0;
}
```

10. Introductin of class in C++ :--

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

class gtu{
    public :    // access specifier
    int a;

    void getdata()
    {
        cout << "Enter marks ";
        cin >> a;
    }

    void printdata()
    {
        cout << "Your marks is: "<<a;
    }
};

int main()
{
    gtu obj;
    obj.getdata();
    obj.printdata();
    return 0;
}
```

11. Write a C++ program to print sum of 2 number using function. (avoid function declaration)

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

// to avoid function delcaration,
   put function defination above main()
int sum (int x, int y){
   int k;
   k = x + y;
   return k;
}
```

```
int main()
{
   int a,b,ans;
   cout<<"Enter 2 values:";
   cin>>a>>b;
   ans = sum (a,b);
   cout<<"answer is : "<<ans<<endl;
   return 0;
}
```

12. Write a C++ program to get percentage from marks of 4 subjects.
Print grade according as per percentage. (above 70 = distinction, 60 to 69 = first class, 50 to 59 = second class, 35 to 49 = pass, below 35 = fail.)

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

int main(){
   int sub1, sub2, sub3, sub4, per;

   cout << "Enter values:" << endl;
   cin>>sub1>>sub2>>sub3>>sub4;

   per = (sub1+sub2+sub3+sub4)/4;
   cout << "Percentage is : " << per << endl;

   if(sub1 >= 35 && sub2 >= 35 && sub3 >= 35 && sub4 >= 35)
   {
      if(per >= 70){
         cout<<"Grade : Distinction"<<endl;
      }
      else if(per >= 60 && per < 70){
         cout<<"Grade : First class"<<endl;
      }
      else if(per >= 50 && per < 60){
         cout<<"Grade : Second class"<<endl;
      }
      else{
         cout<<"Grade : Pass"<<endl;
      }
   }
   else{
      cout<<"Grade : Fail"<<endl;
   }
   return 0;
}
```

13. Intro to Method Overloading :-- program for sum of 2 numbers.

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

// method/function overloading :
   functions with same name but different num of arguments/type of arguments
int sum (int x, int y){
   int k;
   k = x + y;
   return k;
}
double sum (double x, double y){
   double l;
   l = x + y;
   return l;
}

int main()
{
   int a,b,ans1;
   double ans2;
   ans1 = sum (7,5);
   cout<<"answer 1 is : "<<ans1<<endl;
   ans2 = sum (3.2,5.2);
   cout<<"answer 2 is : "<<ans2<<endl;
   return 0;
}
```

14. Find area of circle & rectangle using function overloading. Use function name 'shape'.

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

double shape (int x, double pi = 3.14){
   double k;
   k = double(pi*x*x);
   return k;
}
double shape (double x, double y){
   double l;
   l = double(x * y);
   return l;
}

int main()
{
   int x;
   double ans1;
   double ans2;
   ans1 = shape(2);
   cout<<"Area of circle is : "<<ans1<<endl;
   ans2 = shape(2.5,5.5);
   cout<<"Area of rect is : "<<ans2<<endl;
   return 0;
}
```

15. Find area of triangle, circle, and rectangle using function overloading.

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

float areaOfShape (float x, float y){
    float k;
    k = (0.5 * x * y);
    return k;
}
double areaOfShape (int x, double pi = 3.14){
    double l;
    l = double(pi*x*x);
    return l;
}
double areaOfShape (double x, double y){
    double m;
    m = double(x * y);
    return m;
}
```

```
int main(){
    int x;
    float ans1;
    double ans2;
    double ans3;
    ans1 = areaOfShape(2,2.5);
    cout<<"Area of triangle is : "<<ans1<<endl;
    ans2 = areaOfShape(2);
    cout<<"Area of circle is : "<<ans2<<endl;
    ans3 = areaOfShape(2.5,5);
    cout<<"Area of rectangle is : "<<ans3<<endl;
    return 0;
}
```

16. Write a C++ program to swap two numbers - With function.

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

void swapNum (int a, int b){
    float temp;
    temp = a;
    a = b;
    b = temp;
    cout<<"After swap : "<<"a = "<<a<<", "<<"b = "<<b<<endl;
}

int main(){
    int a,b;
    cout<<"Enter 2 values:";
    cin>>a>>b;
    cout<<"Before swap : "<<"a = "<<a<<", "<<"b = "<<b<<endl;
    swapNum(a,b);
    return 0;
}
```

17. Write a C++ program to perform arithmetic operations on two numbers. (Use switch case).

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

int main()
{
    float a,b,ans;
    char choice;

    cout<<"Enter 2 values : ";
    cin>>a>>b;

    cout<<"Enter choice : ";
    cin>>choice;

    switch (choice)
    {
    case '+':
        ans = a + b;
        cout<<ans<<endl;
        break;

    case '-':
        ans = a - b;
        cout<<ans<<endl;
        break;

    case '*':
        ans = a * b;
        cout<<ans<<endl;
        break;

    case '/':
        ans = a / b;
        cout<<ans<<endl;
        break;

    default:
        cout<<"Wrong input"<<endl;
        break;
    }
}
```

18. Write a C++ program to raise power of n by 3 using function.

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

inline int cube (int x){    // inline --- reduce the exicution time.
    int k;
    k =  x*x*x;
    return k;
}

int main()
{
    int n, ans;
    cout<<"Enter value :"<<endl;
    cin>>n;
    ans = cube(n);
    cout<<"Ans is :"<<ans<<endl;
    return 0;
}
```

19. C++ program for get & print data of 3 employees, using class. (class name - Employee, data members - name, expireance, salary, function name - 1. getInfo, 2. DisplayInfo)

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

class employee{
    public :
    char name[10];
    float years;
    float salary;

    void getInfo()
    {
        cout<<"Employee name: ";
        cin >> name;
        cout<<"Enter Experience(in years) : ";
        cin >> years;
        cout<<"Enter salary : ";
        cin >> salary;
    }

    void displayInfo()
    {
        cout<<"Emplyee name is: "<<name<<endl;
        cout<<name<<" has "<<years<<"years of experience & " <<"his salary is "<<salary<<endl;
    }
};
```

```
int main()
{
    employee obj[3];
    for(int i=0; i<3; i++){
        obj[i].getInfo();
    }
    cout<<""<<endl;
    for(int i=0; i<3; i++){
        obj[i].displayInfo();
    }
    return 0;
}
```

20. Find area of circle using class.

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

class area_of_circle
{
    float r, ans;

public:
    void read()
    {
        cout << "Enter radius : ";
        cin >> r;
    }
    void calculate()
    {
        ans = 3.14 * r * r;
    }
    void show()
    {
        cout << "Area of circle is : " << ans << endl;
    }
};
```

```cpp
int main()
{
    area_of_circle obj;
    obj.read();
    obj.calculate();
    obj.show();
}
```

21. function with arguments in class.

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

class gtu
{
    string name;
    int age;

public:
    void receive(string n, int a)
    {
        name = n;
        age = a;
    }
    void display()
    {
        cout << "Your name is " << name << endl;
        cout << "Your age is " << age << endl;
    }
};
```

```cpp
int main()
{
    gtu obj;
    obj.receive("Dev", 12);
    obj.display();
    return 0;
}
```

22. Introduction to Constructor.

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

class gtu {
    public:
    int a,b;
    gtu(){            // class name and function name same then its called constructor.
        a=10;
        b=20;
    }
    void display(){
        cout<<a<<" and "<<b<<endl;
    }
};

int main() {
    gtu obj;
    obj.display();
    return 0;
}
```

23. Program to understand different types of access specifiers in class.

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

class gtu{
    private:
        int x;

    public:
        int y;
};

int main() {
    gtu obj;
    obj.y = 11;

    cout << obj.y <<endl;

    return 0;
}
```

24. Program to understand types of constructor.

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

class demo
{
    float l, b, ans;

public:
    demo()       // default constructor
    {
        l = 10;
        b = 20;
        ans = l * b;
        cout << "Ans from default : " << ans << endl;
    }

    demo(float x, float y)        // parameterized constructor
    {
        l = x;
        b = y;
        ans = l * b;
        cout << "Ans from parameterized : " << ans << endl;
    }
};
```

```
int main()
{
    demo d1;
    demo d2(4.5, 5);
}
```

25. class name - rectangle, calculate perimeter of rectangle. Use default & parameterized constructor. Use function for calculation of perimeter & display ans.

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

class demo{
    float l,b,ans;
    public:
        demo(){        // default constructor
            l=10;
            b=20;
        }
        demo(float x, float y){     // parameterized constructor
            l=x;
            b=y;
        }

        void displayPerimeter(){
            ans = 2.0*(l+b);
            cout<< "perimeter of rectangle is : "<<ans<<endl;
        }
};
```

```
int main(){
    demo d1;
    d1.displayPerimeter();
    demo d2(4,7);
    d2.displayPerimeter();
}
```

26. Introduction to copy constructor.

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


class demo{
    float l,b,ans;
    public:
        demo(float x, float y){     // parameterized constructor
            l=x;
            b=y;
            cout <<  "l : " << l << " & b : " << b << endl;
        }
        demo(demo & r){     // copy constructor, r = object name
            l=r.l;
            b=r.b;
            cout << "from copy constructor --->> l : " << l << " & b : " << b << endl;

        }
};

int main(){
    demo d1(4,5);
    demo d2(d1);
}
```

27. Constructor overloading.

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

class demo{
    float l,b,ans;
    public:
        demo(float x, float y){     // parameterized constructor
            l=x;
            b=y;
            cout << "l : "<< l << " & b : "<< b << endl;
        }
        demo(float x, float y, float z){     // copy constructor
            l=x;
            b=y;
            ans=z;
            cout << "from copy constructor --->> l : " << l << " & b : " << b <<endl;
            cout << " Ans is : "<<ans<<endl;
        }
};
```

```
int main(){
    demo d1(4,5);
    demo d2(4,5,0);
}
```

28. Introduction to scope resolution operator.

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

int m = 50;
int main(){
    int m =100;
    cout << "value is : " << m << endl;
    cout << "value is : " << ::m << endl;   // access global variable
    return 0;
}
```

29. Use of scope resolution operator with function.

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

class operate{
    public:
    void func();
};

// access function outside the class.
void operate::func(){
    cout << "I'm in class." <<endl;
}
```

```
int main(){
    operate o1;
    o1.func();
    return 0;
}
```

30. Passing object as an argument with constructor.

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

class demo{
    int a;
    public:
    void set(int x){
        a=x;
    }
    void sum(demo o1, demo o2){
        a= o1.a + o2.a;
    }
    void display(){
        cout<<a<<endl;
    }
};
```

```
int main(){
    demo d1, d2, d3;
    d1.set(10);
    d2.set(20);

// pass objects (d1, d2) as an arguments
    d3.sum(d1,d2);
    d1.display();
    d2.display();
    d3.display();
    return 0; 0;
}
```

31. Write a c++ main function to create objects of dist class. Then input two distances & output the sum.

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

class dist
{
    int feet, inch;

    public:
    void set(int f, int i)
    {
        feet = f;
        inch = i;
    }

    void sum(dist o1, dist o2)        // copy constructor
    {
        inch = o1.inch + o2.inch;

        // inch to feet conversion
        feet = inch / 12;
        inch = inch % 12;
        feet = o1.feet + o2.feet + feet;
    }

    void display()
    {
        cout << "Distance : " << feet << " feet &  " << inch << “inch.” << endl;
    }
};

int main()
{
    dist d1, d2, d3;
    d1.set(5, 10);
    d2.set(6, 12);
    d3.sum(d1, d2);
    d1.display();
    d2.display();
    d3.display();
}
```

32. Write a c++ main function to create objects of hrs class then input two time & output the sum.

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

class hrs
{
    int hours, minutes;

public:
    void set(int h, int m)
    {
        hours = h;
        minutes = m;
    }

    void sum(hrs o1, hrs o2)          // copy constructor
    {
        minutes = o1.minutes + o2.minutes;

        // minutes to hour conversion
        hours = minutes / 60;
        minutes = minutes % 60;
        hours = o1.hours + o2.hours + hours;
    }

    void display()
    {
        cout << "Total hourss : " << hours << " & minutes : " << minutes << endl;
    }
};

int main()
{
    hrs d1, d2, d3;
    d1.set(5, 30);
    d2.set(3, 60);
    d3.sum(d1, d2);
    d1.display();
    d2.display();
    d3.display();
}
```

33. Write a C++ program to find minimum of two given numbers using inline function.(ternary operator)

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

inline int showMin(int a, int b){
    int ans;
    ans =  a<b ? a:b;
    return ans;
}

int main(){
    int a, b, ans;
    cout<<"Enter a numbers : ";
    cin >> a >> b;
    ans = showMin(a, b);
    cout<<"Min num is : "<<ans << endl;
    return 0;
}
```

34. Create a class 'Student' with three data members which are name, age & address. The constructor of the class assigns default values to name as "unknown", age as '0' & address as "not available". It has two functions setinfo and getinfo to read and print these parameters.

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

class Student{
    public:
    string name;
    int age;
    string address;

    Student(){
        name="unknown";
        age=0;
        address="not available";
    }

    void setInfo(string n, int a, string h){
        name = n;
        age = a;
        address = h;
    }

    void displayInfo(){
        cout<<"Student name: "<<name<<" age: "<<age<<" address : "<<address<<endl;
    }
};
```

```cpp
int main(){
    Student s1;
    s1.displayInfo();
    s1.setInfo("Dev", 25, "xyz");
    s1.displayInfo();
}
```

35. Create a class to print the area of a square and a rectangle. The class has two functions with the same name but different number of parameters. The function for printing the area of rectangle has two parameters which are its length and breadth respectively while the other function for printing the area of square has one parameter which is the side of the square.

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

class AreaOfShape {
    float l, b, ans;

public:
    void showAreaofShape(float x, float y) {
        l = x;
        b = y;
        ans = float(x * y);
        cout << "Area of rect is : " << ans << endl;
    }
    void showAreaofShape(float x) {
        l = x;
        ans = float(x * x);
        cout << "Area of rect is : " << ans << endl;
    }
};
```

```
int main()
{
    AreaOfShape a1;
    a1.showAreaofShape(2, 5);
    a1.showAreaofShape(2);
    return 0;
}
```

36. Write a c++ program to calculate multiplication of two numbers and three numbers using constructor overloading.

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

class multi {
    int a, b, c, ans;

    public:
    multi(int x, int y){
        a = x;
        b = y;
        ans = x * y;
    }
    multi(int x, int y, int z){
        a = x;
        b = y;
        c = z;
        ans = x * y * z;
    }

    void displayAns(){
        cout << "Ans of multiplication of values is : " << ans << endl;
    }
};
```

```
int main()
{
    multi c1(2, 5);
    multi c2(2, 5, 6);
    return 0;
}
```

37. Introduction to friend function.

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

class test{
    private:
        int a;
    protected:
        int b;
    public:
        int c;

    friend void func();
};

void func() {
    test t;
    t.a = 10;
    t.b = 20;
    t.c = 30;
    cout << t.a << " " << t.b << " " << t.c << endl;
};

int main() {
    func();
    return 0;
}
```

38. Introduction to destrucotr.

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

class student {
 public:
 student() {              // constructor
    cout<< "find";
 }

 ~student() {            // destructor
    cout<< "course";
 }
};

int main() {
 student obj;
 return 0;
}
```

39. Write a C++ program to find volume of cube, cylinder and rectangular box using concepts of function overloading.

```
#include <iostream>
using namespace std;
#define pi 3.14

class Vol{
    int l, b, h, r;
    float ans;
    public:

    float volOfShape(int x, int y){
        r=x;
        h=y;
        ans = pi*r*r*h;
        return ans;
    }

    float volOfShape(float x){
        l=x;
        ans = l*l*l;
        return ans;
    }

    float volOfShape(float x, float y, float z){
        l=x;
        b=y;
        h=z;
        ans = l*b*h;
        return ans;
    }

    void displayVol(){
        cout<<"Volume of shape is : "<<ans<<endl;
    }

};

int main(){
    Vol v1, v2, v3;
    v1.volOfShape(2,4);
    v1.displayVol();
    v2.volOfShape(3);
    v2.displayVol();
    v3.volOfShape(3,4,5);
    v3.displayVol();
}
```

40. Write a c++ program to define a class 'employee' to store records of 4 employees of company. including following members: Data member: (1) Emp_id (2) Emp_name (3) Emp_dept Member Functions: (1)getdata() (2)displaydata().

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

class employee
{
    string Emp_name;
    int Emp_id;
    string Emp_dept;

public:
    void getData()
    {
        cout << "Emp id : ";
        cin >> Emp_id;
        cout << "Emp name : ";
        cin >> Emp_name;
        cout << "Emp dept : ";
        cin >> Emp_dept;
    }
    void displayData()
    {
        cout << "Employee id : " << Emp_id << endl;
        cout << "Employee name : " << Emp_name << endl;
        cout << "Department : " << Emp_dept << endl;
        cout << "---------------" << endl;
    }
};

int main()
{
    employee Emp[4];
    for (int i = 0; i <= 3; i++)
    {
        Emp[i].getData();
    }
    cout << " " << endl;
    for (int i = 0; i <= 3; i++)
    {
        Emp[i].displayData();
    }
    return 0;
}
```

41. Find mean value of 2 numbers using friend function.

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

class base {
    int val1, val2;

    public:
    void getData() {
        cout << "Enter values:";
        cin >> val1>>val2;
    }
    friend float meanVal(base b1);

};

float meanVal(base b1) {
    cout << b1.val1 << " " << b1.val2 << endl;
    return (float(b1.val1 + b1.val2) / 2);
}

int main() {
    base obj;
    obj.getData();
    float res = meanVal(obj);
    cout << "\n Mean value is : " << res;

    return 0;
}
```

42. Introduction to inheritance :-- Single level inheritance.

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

class A{
    public:
    void display(){
        cout << "University" << endl;
    }
};

class B: public A{
    public:
    void message(){
        cout << "Collage" << endl;
    }
};
```

```
int main(){
    B obj;
    obj.display();
    obj.message();
    return 0;
}
```

43. multi level inheritance :--

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

class A{
    public:
    void display(){
        cout << "University" << endl;
    }
};

class B: public A{
    public:
    void message(){
        cout << "Collage" << endl;
    }
};

class C: public B{
    public:
    void show(){
        cout << "Department" << endl;
    }
};
```

```
int main(){
    C obj;
    obj.display();
    obj.message();
    obj.show();
    return 0;
}
```

44. Write a program for sum of 2 values using multi level inheritance. Use grandParent class to get 1st value, parent class to get 2nd value & child class to do sum of them & display it.

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

class AA{
    public:
    int val1;
    void getData1(){
        cout << "Enter 1st val : ";
        cin >> val1;
    }
};

class BB: public AA{
    public:
    int val2;
    void getData2(){
        cout << "Enter 2nd val : ";
        cin >> val2;
    }
};
```

```
class CC: public BB{
    public:
    void add(){
        cout << "-------------------- "<< endl;
        cout << "Sum is : "<< (val1+val2) << endl;
    }
};

int main(){
    CC obj;
    obj.getData1();
    obj.getData2();
    obj.add();
    return 0;
}
```

45. Create three class customer, depositor and borrower. Design a base class customer (name, mobile). Derive a class depositor (acc_no, balance) from customer class. Derive a class borrower (loan_no, loan_amount) from depositor class. Write a C++ program using necessary member functions to read and display the details of customer.

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

class Customer{
   public:
   string name, mobile_num;
   void getCustDetails(){
      cout << "Enter name : ";
      cin >> name;
      cout << "Enter mobile number : ";
      cin >> mobile_num;
   }
   void showCustDetails(){
      cout << "Customer name : " << name << endl;
      cout << "Mobile : " << mobile_num << endl;
   }
};

class Depositor: public Customer{
   public:
   string acc_num;
   float acc_bal;
   void getAccDetails(){
      cout << "Enter Acc num : ";
      cin >> acc_num;
      cout << "Enter Balance : ";
      cin >> acc_bal;
   }
   void showAccDetails(){
      cout << "Acc : " << acc_num << endl;
      cout << "Balance : " << acc_bal << endl;
   }
};

lass Borrower: public Depositor{
   public:
   string loan_num;
   float loan_amount;
   void getLoanDetails(){
      cout << "Enter loan number : ";
      cin >> loan_num;
      cout << "Enter loan amount : ";
      cin >> loan_amount;
   }
   void showLoanDetails(){
      cout << "Loan number : " << loan_num << endl;
      cout << "Loan amount : " << loan_amount << endl;
   }
};

int main(){
   Borrower obj;
   obj.getCustDetails();
   obj.getAccDetails();
   obj.getLoanDetails();
   cout << "-----------------------" << endl;
   obj.showCustDetails();
   obj.showAccDetails();
   obj.showLoanDetails();
   return 0;
}
```

46. Multiple inheritance :--

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

class A{
    public:
    void display(){
        cout << "University" << endl;
    }
};

class B{
    public:
    void show(){
        cout << "Engineering" << endl;
    }
};
```

```
class C: public A, public B{
    public:
    void message(){
        cout << "SURAT" << endl;
    }
};

int main(){
    C obj;
    obj.display();
    obj.show();
    obj.message();
    return 0;
}
```

47. Design 2 base classes, "Personal"(name, id, Birthdate), "Academic"(10th Marks, 12th marks), Derive a class "biodata" from both these classes. & display the data.

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

class Personal{
    public:
    string name, birthdate;
    int id;
    void getPersonalDetails(){
        cout << "Enter id: ";
        cin >> id;
        cout << "Enter name: ";
        cin >> name;
        cout << "Enter B.O.D.: ";
        cin >> birthdate;
    }
};

class Academic{
    public:
    int sscMarks, hscMarks;
    void getMarks(){
        cout << "Enter ssc marks : ";
        cin >> sscMarks;
        cout << "Enter hsc marks : ";
        cin >> hscMarks;
    }
};
```

```
class Biodata: public Personal, public Academic{
  public:
  void ShowBio(){
    cout << "ID : " << id << endl;
    cout << "Name : " << name << endl;
    cout << "B.O.D. : " << birthdate << endl;
    cout << "SSC Marks : " << sscMarks << endl;
    cout << "HSC Marks : " << hscMarks << endl;
  }
};

int main(){
    Biodata obj;
    obj.getPersonalDetails();
    obj.getMarks();
    cout << "--------BIODATA-----------" << endl;
    obj.ShowBio();
    return 0;
}
```

48. hierarchical inheritance :--

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

class A{
    public:
    void display(){
        cout << "From A (parent)" << endl;
    }
};

class B:public A{
    public:
    void show(){
        cout << "From B (first child)" << endl;
    }
};

class C:public A{
    public:
    void provide(){
        cout << "From C (second child)" << endl;
    }
};

int main(){
    C obj;
    obj.display();
    obj.provide();
    cout << "----------------" << endl;
    B obj11;
    obj11.display();
    obj11.show();
}
```

49. Make class with name "Bank". Its data members are Acc_num & name. Create 2 classes from "Bank" 1) "savings", 2) "current". Write a function "getBal" in which ask for balance & if its less than 2000 in saving acc, then display "insufficient balance". Write a function "getBal" in which ask for balance & if its less than 5000 in current account, then display "insufficient balance".

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

class Bank{
    public:
    int acc_num;
    string name;
    void getCustomerDetails(){
        cout<<"Enter Acc Name : ";
        cin >> name;
        cout<<"Enter Acc number : ";
        cin >> acc_num;
    }
    void showCustomerDetails(){
        cout<<"Acc holder name : "<<name<<endl;
        cout<<"Acc number : "<<acc_num<<endl;
    }
    void CheckBal(float avil_bal, int min_bal){
        if (avil_bal < min_bal)
        {
            cout<<"Acc has insufficient balance"<<endl;
        }else{
            cout<<"Acc balance is : "<<avil_bal<<endl;
        }
    }
};

class Saving_Acc:public Bank{
    public:
    float saving_acc_bal;
    void getBal(){
        cout<<"Enter Saving account balance : ";
        cin >> saving_acc_bal;

        Bank::CheckBal(saving_acc_bal, 2000);
    }
};

class Current_Acc:public Bank{
    public:
    float current_acc_bal;
    void getBal(){
        cout<<"Enter current account balance : ";
        cin >> current_acc_bal;

        Bank::CheckBal(current_acc_bal, 5000);
    }
};
```

```
int main(){
    Current_Acc obj;
    obj.getCustomerDetails();
    obj.showCustomerDetails();
    cout<<"--------------------"<<endl;
    obj.getBal();
    cout<<"--------------------"<<endl;
    Saving_Acc obj1;
    obj1.getBal();

    return 0;
}
```

50. write a program to calculate area and perimeter of rectangle using heirarchical inheritance.

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

class Rectangle {
    public:
    int wid, len;
    void getData(){
        cout << "Enter width : ";
        cin >> wid;
        cout << "Enter length : ";
        cin >> len;
    }
};

class areaOfRect : public Rectangle{
    public:
    int ans1;
    void area(){
        ans1 = wid * len;
        cout << "Area of rectangle : " << ans1 << endl;
    }
};

class perimOfRect : public Rectangle{
    public:
    int ans2;
    void perim(){
        ans2 = 2 * (wid + len);
        cout << "Perimeter of rectangle : " << ans2 << endl;
    }
};

int main(){
    areaOfRect a1;
    a1.getData();
    cout << "--------------------------------" << endl ;
    a1.area();

    perimOfRect p2;
    p2.getData();
    cout << "--------------------------------" << endl ;
    p2.perim();

    return 0;
}
```

51. Hybrid inheritance :--

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

class A{
    public:
    void print(){
        cout<<"From class A"<<endl;
    }
};
class B:public A{
    public:
    void show(){
        cout<<"From class B"<<endl;
    }
};
class C:public B{
    public:
    void display(){
        cout<<"From class C"<<endl;
    }
};
class D:public B{
    public:
    void print_last(){
        cout<<"From class D"<<endl;
    }
};
```

```cpp
int main(){
    C obj;
    cout<<"------------------------"<<endl;
    obj.print();
    obj.show();
    obj.display();

    D obj1;
    cout<<"------------------------"<<endl;
    obj1.show();
    obj1.print_last();
    return 0;
}
```

52. understand ambiguity & virtual keyword.

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

class A {
public:
    void show()
    {
        cout << "You are in exam room \n";
    }
};

class B : public A {
};

class C : public A {
};

class D : public virtual B, public virtual C {
};
```

```cpp
int main()
{
    D obj;
    obj.B::show();

    return 0;
}
```

53. Create 4 classes student (student_id, name), theory_marks (theory marks), practical_marks (practical marks) & result (total, average). Here, theory_marks & practical_marks classes are derived from the student class & the result class is derived from the class theory_marks and practical_marks. Write a C++ program using necessary member functions to read & display the result of 3 students.

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

class Student{
  protected:
  int student_id;
  string name;

  public:
  Student(int id, const string &n) : student_id(id), name(n) {}

  void display(){
    cout << "Student ID: " << student_id << endl;
    cout << "Name: " << name << endl;
  }
};

class TheoryMarks : public Student{
  protected:
  int theory_marks;

  public:
  TheoryMarks(int id, const string &n, int t_marks) : Student(id, n), theory_marks(t_marks) {}

  void display(){
    Student::display();
    cout << "Theory Marks: " << theory_marks << endl;
  }
};

class PracticalMarks : public Student{
  protected:
  int practical_marks;

  public:
  PracticalMarks(int id, const string &n, int p_marks) : Student(id, n), practical_marks(p_marks) {}

  void display(){
    Student::display();
    cout << "Practical Marks: " << practical_marks << endl;
  }
};
```

rest in next page...

53. Create 4 classes student (student_id, name), theory_marks (theory marks), practical_marks (practical marks) & result (total, average). Here, theory_marks & practical_marks classes are derived from the student class & the result class is derived from the class theory_marks and practical_marks. Write a C++ program using necessary member functions to read & display the result of 3 students.

from previous page.......

```
class Result : public TheoryMarks, public PracticalMarks{
  public:
  Result(int id, const string &n, int t_marks, int p_marks) : TheoryMarks(id, n, t_marks), PracticalMarks(id, n, p_marks) {}

  void display(){
    TheoryMarks::display();
    PracticalMarks::display();
    cout << "Total Marks: " << theory_marks + practical_marks << endl;
    cout << "Average Marks: " << (theory_marks + practical_marks) / 2.0 << endl;
  }
};

int main(){
  Result student1(1, "sam", 80, 90);
  Result student2(2, "adi", 75, 85);
  Result student3(3, "hunaif", 85, 95);

  cout << "Result for Student 1:" << endl;
  student1.display();
  cout << endl;

  cout << "Result for Student 2:" << endl;
  student2.display();
  cout << endl;

  cout << "Result for Student 3:" << endl;
  student3.display();
  cout << endl;

  return 0;
}
```

54. Method overriding :---->>

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

class base{
    public:
    void display(){
        cout << "Function of base class" << endl;
    }
};

class derived: public base{
    public:
    void display(){
        cout << "Function of derived class" << endl;
    }
};

int main(){
    derived d1, d2;
    d1.display();
    d2.base::display();     // access overridden function of base class
    return 0;
}
```

55. Write a C++ program to find area of rect(base class), area of triangle(derived class) using function overriding.

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

class Rect{
    public:
    int l, b, ans;

    void AreaOfShape(int l, int b){
        ans = l*b;
        cout <<"Area of rectanngle is : "<< ans << endl;
    }
};
class Triangle:public Rect{
    public:
    int h, b;
    float ans1;

    void AreaOfShape(float h, float b){
        ans1 = 0.5*b*h;
        cout << "Area of triangle is : "<< ans1 << endl;
    }

};
```

```
int main(){
    Triangle t1, t2;
    t1.AreaOfShape(2,5);
    t2.Rect::AreaOfShape(3,5);
    return 0;
}
```

56. Introduction to this keyword.

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

class base{
    int val;
    public:
    void set(int val){
      this->val = val;
    }

    void show(){
        cout <<"Value is : "<< val << endl;
    }
};

int main(){
    base obj;
    obj.set(50);
    obj.show();
    return 0;
}
```

57. Structure & string copy function

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

int main(){
    struct student{     // structure
        int num;
        char name[25];
    };

    student stu;
    stu.num = 123;

    strcpy(stu.name, "raj");    // copy string

    cout << stu.num << endl;
    cout << stu.name << endl;
    return 0;
}
```

58. Program to get the reverse of the string.

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


int main(){
    char name[25];
    cout << "Enter name : ";
    cin >> name;

    strrev(name);
    cout << name << endl;

    return 0;
}
```

59. Introduction to manipulation operator (setw).

```
#include <iostream>
#include <iomanip>

using namespace std;

int main(){
    int a=3, b=78, c=12334;

    cout<<"value of a without setw : "<< a <<endl;
    cout<<"value of b without setw : "<< b <<endl;
    cout<<"value of c without setw : "<< c <<endl;
    cout<<"-----------------------------------"<<endl;
    cout<<"value of a : "<< setw(4) << a <<endl;
    cout<<"value of b : "<< setw(4) << b <<endl;
    cout<<"value of c : "<< setw(4) << c <<endl;

    return 0;
}
```

60. Program to create file.

```
#include <iostream>
#include <fstream>

using namespace std;

int main(){
    fstream new_file;

    new_file.open("gtu.txt", ios::out);
    // out - start writing in file from the start / overwrite in file.  in- for read, out - for write.

    if (!new_file){
        cout << "File creation failed" << endl;
    }
    else{
        cout << "File created successfully" << endl;
        new_file.close();
    }

    return 0;
}
```

61. Program to create file. & write into it.

```
#include <iostream>
#include <fstream>

using namespace std;

int main(){
    fstream new_file;

    new_file.open("abc.txt", ios::app);     // app = append - start writing in file at the end of file.

    if (!new_file){
        cout << "File creation failed" << endl;
    }
    else{
        cout << "File created successfully" << endl;
        new_file << "This is a file created by C++ program" << endl;
        new_file.close();
    }

    return 0;
}
```

62. Program to read from file.

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

int main(){
    fstream new_file;
    new_file.open("gtu.txt", ios::in);

    if (!new_file){
        cout << "File not found" << endl;
    }
    else{
        char ch;
        while(1){
            new_file >> ch;
            if (new_file.eof())        // eof = end of file
            break;

            cout << ch << endl;
        }
    }
    new_file.close();
    return 0;
}
```

63. Develop a C++ program that creates a new file,
Write “Computer Engineering” message in that file and display contents of that file on the screen.

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

int main(){
    fstream newFile;

    newFile.open("fields.txt", ios::out);

    if(!newFile){
        cout << "File creation failed" << endl;
    }
    else{
        cout << "File created successfully" << endl;
        newFile << "Computer Engineering" << endl;
        newFile.close();
    }

    newFile.open("fields.txt", ios::in);
    if(newFile){
        char ch;
        while(1){
            newFile >> ch;
            if (newFile.eof())        // eof = end of file
            break;

            cout << ch << endl;
        }
    }
    else{
        cout << "File not found" << endl;
    }
    newFile.close();
    return 0;
}
```

64. Write a program to copy contents of gtu.txt into su.txt

this program has an issue of ignore space while copying

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

int main(){
    fstream newFile1, newFile2;
    char ch;

    newFile1.open("gtu.txt", ios::in);
    newFile2.open("su.txt", ios::out);

    if(newFile1){
        while(1){
            newFile1 >> ch;

            if(isspace(ch)){
                cout << "there is space here." << endl;
            }
            else {
                cout << "copying..." << ch << endl;
            }

            if(newFile1.eof())
            break;

            newFile2 << ch;
        }
    }
    else{
        cout << "File does not exist" << endl;
    }

    newFile1.close();
    newFile2.close();

    return 0;
}
```

to avoid the issue of ignoring space while copy use “getline” function.

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

int main(){
    fstream newFile1, newFile2;
    string line;

    newFile1.open("gtu.txt", ios::in);
    newFile2.open("su.txt", ios::out);

    if(newFile1){
        while(getline(newFile1, line)){
        // getline function read whole line.
            newFile2 << line << "\n";
        }
    }
    else{
        cout << "File does not exist" << endl;
    }

    newFile1.close();
    newFile2.close();

    return 0;
}
```

65. write a C++ program to create a file with following pattern in it.

```
*
* *
* * *
```

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

int main(){
    int i,j;
    string line;
    fstream new_file;

    new_file.open("pattern1.txt", ios::out);

    if (!new_file){
        cout << "File creation failed" << endl;
    }
    else{

        for(i = 1; i <= 3; i++)
        {
            for(j = 1; j <= i; j++)
            {
                new_file << "* ";
            }
            new_file << endl;
        }

        cout << "File created successfully" << endl;
        new_file.close();
    }

    return 0;
}
```

66. Operator overloading :-

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

class testclass{
    int count;

    public:
    testsclass(){
        count = 5;
    }

    void operator --(){
        count = count - 2;
    }
    void display(){
        cout << count << endl;
    }
};
```

```
int main(){
    testclass t;
    --t;
    t.display();

    return 0;
}
```

67. Introduction to Virtual funtion :-

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

class base {
    public:
    virtual void print(){
        cout << "print base class\n";
    }
    void show(){
        cout << "show base class\n";
    }
};

class derived : public base {
    public:
    void print(){
        cout << "print derived class\n";
    }
    void show(){
        cout << "show derived class\n";
    }
};
```

```
int main(){
    base* bptr;
    // int a ---> normal variable
    // int *a ---> pointer variable
    derived d;
    bptr = &d;

    // Virtual function, binded at runtime
    bptr->print();

    // Non-virtual function, binded at compile time
    bptr->show();
    return 0;
}
```

68. Introduction to abstract class ;-

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

class shape{
    public:
    float dimension;

    public:
    void getdimension(){
        cin >> dimension;
    }

    virtual float area() = 0;
};

class square: public shape{
    public:
    float area(){
        return dimension * dimension;
    }
};

class circle: public shape{
    public:
    float area(){
        return 3.14 * dimension * dimension;
    }
};

int main(){
    square s;
    circle c;

    cout << "Enter dimention for square : ";
    s.getdimension();

    cout << "Area of square is " << s.area() << endl;

    cout << "Enter dimension for circle : ";
    c.getdimension();

    cout << "Area of circle is " << c.area() << endl;

    return 0;
}
```

69. Make a class Shape with data member int dimension and member function calculate_Area(). Make this class abstract. Override this method in class circle for calculating area of circle.

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

class shape{
    public:
    float dimension;

    public:
    void get_dimension(){
        cout << "Enter dimension for circle : ";
        cin >> dimension;
    }

    virtual float calculate_Area() = 0;
};

class circle: public shape{
    public:
    float calculate_Area(){
        cout << "Area of circle is " <<  3.14 * dimension * dimension << endl;
    }
};

int main(){
    circle circ;

    circ.get_dimension();
    circ.calculate_Area();

    return 0;
}
```

70. Write a program where you define a class A. which takes two variables int a & int b. Class B inherits Class A and adds two variable defined in class A. Make function get_data and Print_data in class A. Define function Add in class B.

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

class A{
    public:
    int a, b, ans;

    public:
    void get_data(){
        cout << "Enter value : ";
        cin >> a >> b;
    }
    void Print_data(){
        cout << "sum of value is : " << ans << endl;
    }
};

class B : public A{
    public:
    void Add(){
        ans = a + b;
    }
};
```

```
int main(){
    B b1;
    b1.get_data();
    b1.Add();
    b1.Print_data();
}
```

71. Introduction to tatic data members...

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

class item{
    static int count;

    public:
    void get_data(){
        count++;
    }
    void getcount(){
        cout << "count is" << count << endl;
    }
};

int item::count = 10;
```

```
int main(){
    item a,b;
    a.getcount();
    b.getcount();

    a.get_data();
    b.get_data();

    cout << "------------------" << endl;

    a.get_data();
    b.get_data();
    return 0;
}
```

72. Introduction to tatic function..

```
// static function...

#include <iostream>
using namespace std;

class test{
    int code;
    static int count;

    public:
    void setcode(){
        code = ++count;
    }
    static void showcount(){
        cout << "count is " << count << endl;
    }
};

int test::count;
```

```
int main(){
    test t1, t2, t3;
    t1.setcode();
    t2.setcode();

    test::showcount();

    cout << "------------------" << endl;

    t3.setcode();
    test::showcount();
    return 0;
}
```