INDEX

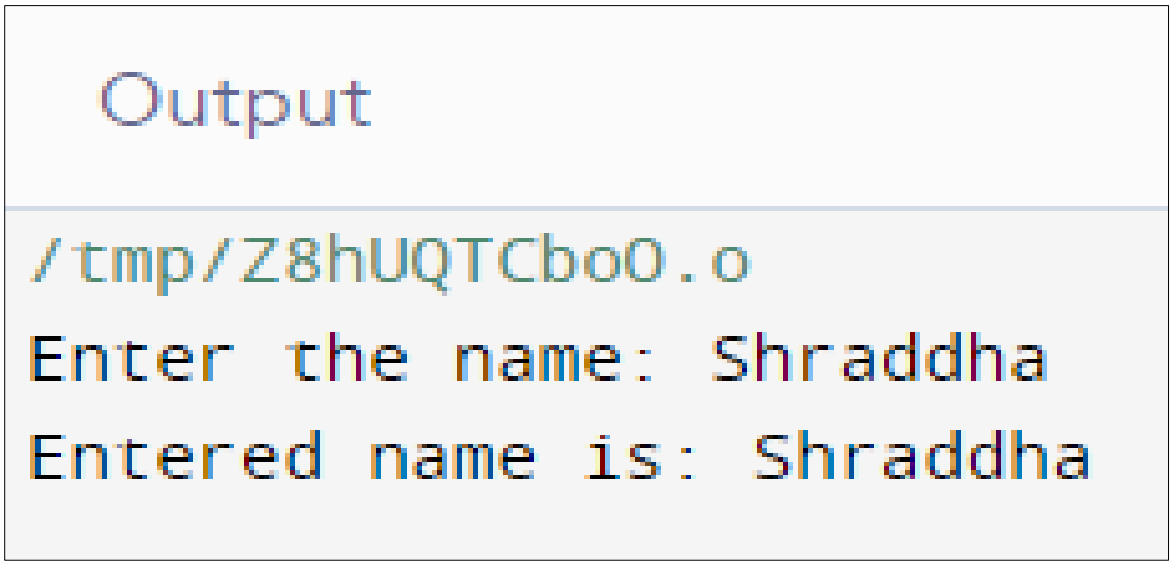
Practical 1

C++ program to print name.

CODE:-

```
// C++ program to print name
// as output
#include <iostream>
using namespace std;

// Driver code
int main()
{
  string name;

  cout << "Enter the name: ";
  cin >> name;

  cout << "Entered name is: " <<
          name;
    return 0;
}
```

OUTPUT:-



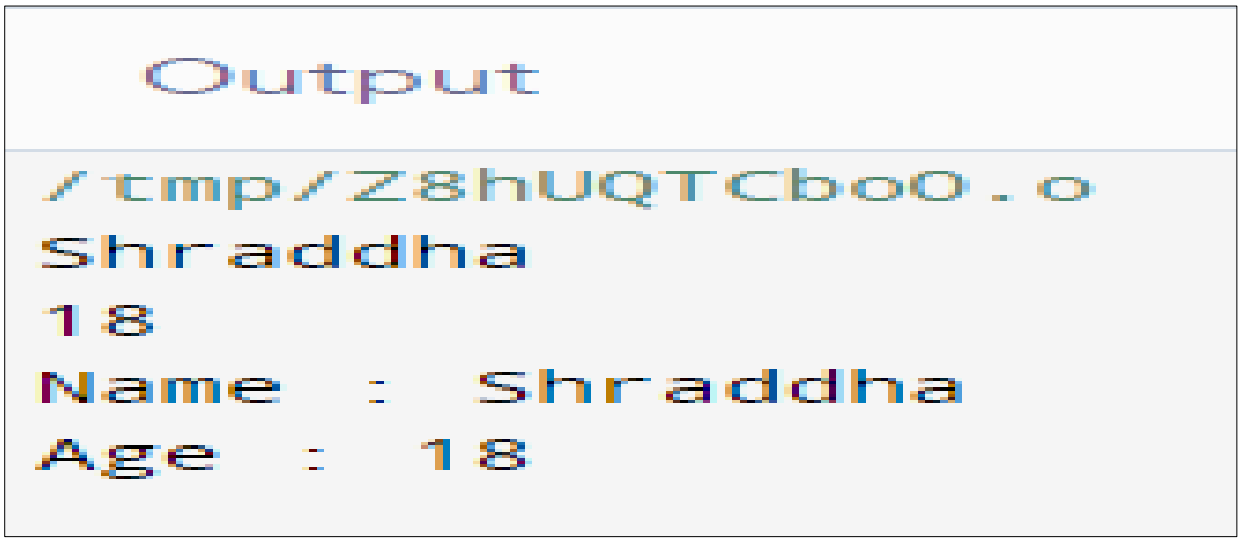
Practical 2

C++ Programs to Demonstrate the Taking Multiple Inputs from the User.

CODE:-

```
// C++ program to demonstrate the taking
// multiple inputs from the user
#include <iostream>
using namespace std;

// Driver Code
int main()
{
    string name;
    int age;

    // Take multiple input using cin
    cin >> name >> age;

    // Print output
    cout << "Name : " << name << endl;
    cout << "Age : " << age << endl;

    return 0;
}
```

OUTPUT:-



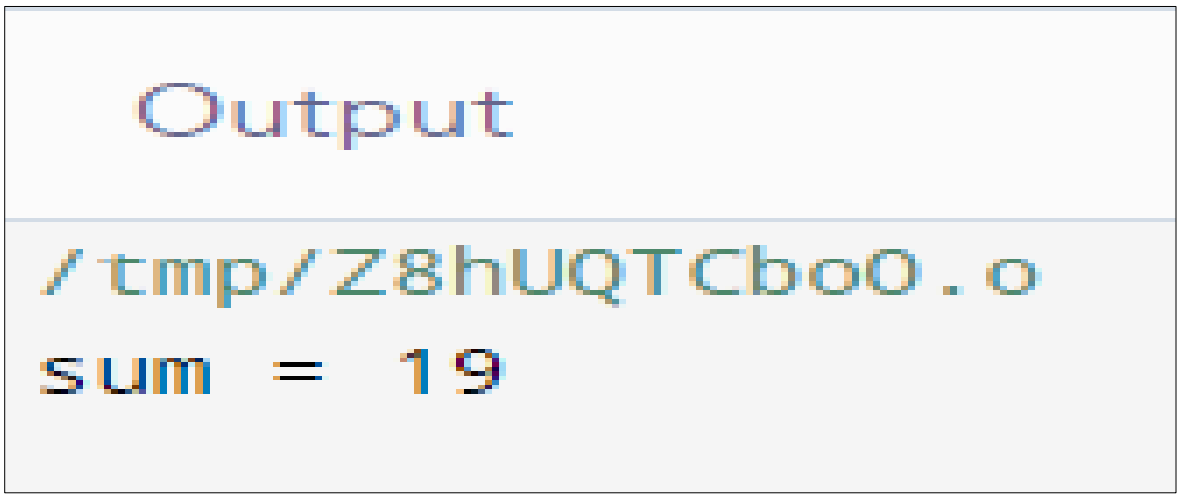
Practical 3

C++ programs to add two Number using Addition Operator.

CODE:-

```
// C++ program to add two number
// using addition operator
#include <iostream>
using namespace std;

// Function to return sum
// of two number
int addTwoNumber(int A, int B)
{
    // Return sum of A and B
    return A + B;
}

// Driver Code
int main()
{
    // Given two number
    int A = 4, B = 11;

    // Function call
    cout << "sum = " << addTwoNumber(A, B);
    return 0;
}
```

OUTPUT:-



Practical 4

C++ program to illustrate the floting point literals.

CODE:-

```
//C++ program to illustrate the floting point literals
#include<iostream>
using namespace std;
int main()
{
   //float literal
   float e=2.7f;
   //double literal
   double pi=3.14;
   //long double literal
   long double g=9.8l;
   cout<<"The value of Pi:"<<pi<<endl;
   cout<<"The value of e:"<<e<<endl;
   cout<<"The value of g:"<<g<<endl;
   return 0;
}
```

OUTPUT:-

```
Output

/tmp/6oz7ORrHRs.o
The value of Pi:3.14
The value of e:2.7
The value of g:9.8


=== Code Execution Successful ===
```

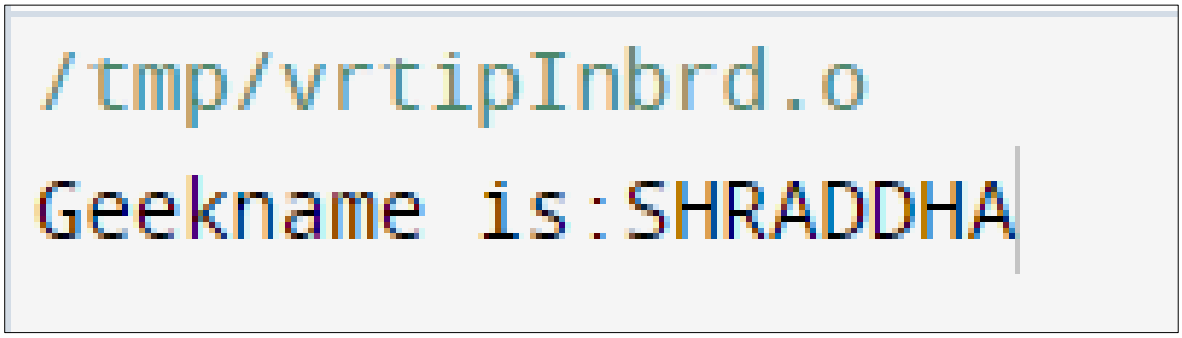
Practical 5

C++ program to demonstrate accessing of data members.

CODE:-

```
#include <bits/stdc++.h>
using namespace std;
class Geeks {
   //Access specifier
public:
   //Data Members
   string geekname;
   //Member Function()
   void printname() { cout << "Geekname is:" << geekname; }
};
int main()
{
   //declare an object of class geeks
   Geeks obj1;
   //Accessing data member
   obj1.geekname= "SHRADDHA";
   //accessing member function
   obj1.printname();
   return 0;
}
```

OUTPUT:-



Practical 6

C++ program to demonstrate function declare outside class.

CODE:-

```
// C++ program to demonstrate function
// declaration outside class
#include <bits/stdc++.h>
using namespace std;
class Geeks
{
   public:
   string geekname;
   int id;
   // printname is not defined inside class definition
   void printname();

   // printid is defined inside class definition
   void printid()
   {
      cout <<"Geek id is: "<<id;

   }
};

// Definition of printname using scope resolution operator ::
void Geeks::printname()
{
   cout <<"Geekname is: "<<geekname;
}
int main() {
   Geeks obj1;
   obj1.geekname= "xyz";
   obj1.id=15;

   // call printname()
   obj1.printname();
   cout << endl;

   // call printid()
   obj1.printid();
   return 0;
}
```

OUTPUT:-

```
/tmp/vrtipInbrd.o
Geekname is: xyz
Geek id is: 15
```

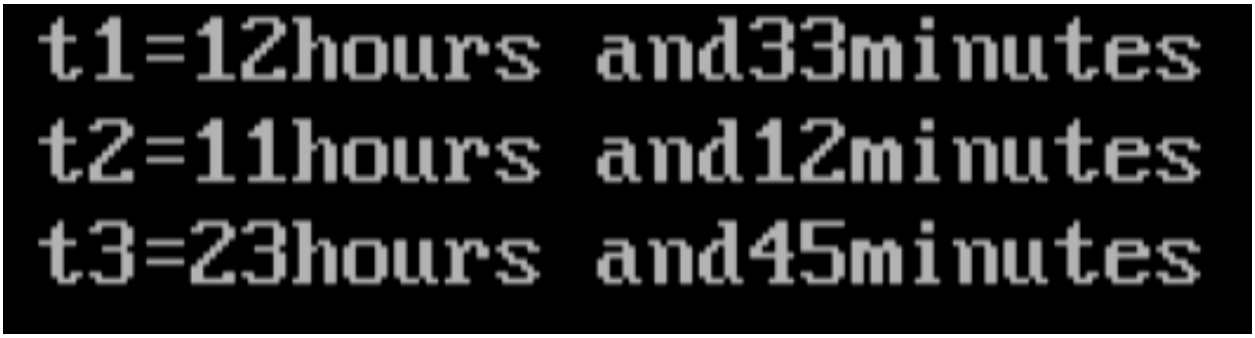
Practical 7

Design a class "time" with hours and member and to get and print data of time class also design a sum() with object as argument to add two objects of time class.

CODE:-

```
#include<iostream.h>
#include<conio.h>
class time
{
int hours;
int minutes;
public:
void gettime(int h, int m)
{
hours=h;
minutes=m;
}
void puttime()
{
cout<<hours<<"hours and";
cout<<minutes<<"minutes"<<"\n";
}
void sum(time,time);
};
void time::sum(time t1, time t2)
{
minutes=t1.minutes + t2.minutes;
hours= minutes/60;
minutes= minutes%60;
hours= hours + t1.hours + t2.hours;
}
void main()
{
clrscr();
time t1,t2,t3;
t1.gettime(12,33);
t2.gettime(11,12);
t3.sum(t1,t2);
cout<<"t1=";
t1.puttime();
cout<<"t2=";
t2.puttime();
cout<<"t3=";
t3.puttime();
getch();
}
```

OUTPUT:-


t1=12hours and33minutes
t2=11hours and12minutes
t3=23hours and45minutes

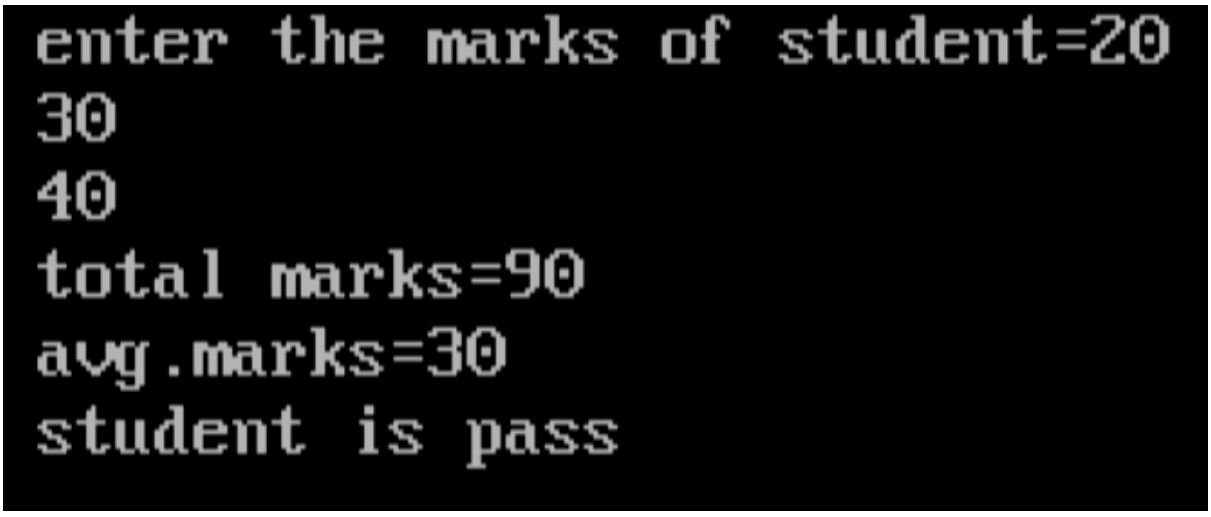
Practical 8

Write a cpp program which accept marks of 3 subjects.Calculate total & Average marks and also check student is fail or pass.(if avg.above or equal to 50 the,,pass").

CODE:-

```
#include<iostream.h>
#include<conio.h>
void main()
{
int a,b,c,total;
float avg;
clrscr();
cout<<"enter the marks of student=";
cin>>a>>b>>c;
total=(a+b+c);
avg=(a+b+c)/3;
cout<<"total marks="<<total<<endl;
cout<<"avg.marks="<<avg<<endl;
if(total>=35)
{
cout<<"student is pass"<<endl;
}
else
{
cout<<"student is fail";
}
getch();
}
```

OUTPUT:-



Practical 9

Design a class "employee" with appropriate members. demonstrate array of objects.

CODE:-

```
#include<iostream.h>
#include<conio.h>
class employee
{
char name[30];
float age;
public:
void getdata();
void putdata();
};
void employee::getdata()
{
cout<<"enter name=";
cin>>name;
cout<<"enter age=";
cin>>age;
}
void employee::putdata()
{
cout<<"name:"<<name<<"\n";
cout<<"age:"<<age<<"\n";
}
const int size=3;
void main()
{
clrscr();
int i;
employee manager [size];
for(i=0;i<size;i++)
{
cout<<"\n details of manager"<<i+1<<"\n";
manager[i].getdata();
}
cout<<"\n";
for(i=0;i<size;i++)
{
cout<<"\n manager"<<i+1<<"\n";
manager[i].putdata();
}
getch();
}
```

OUTPUT:-

```
 details of manager1
enter name= Poonam
enter age= 19.5

 details of manager2
enter name= Sadaf
enter age= 18

 details of manager3
enter name= Shradha
enter age= 18


 manager1
name:Poonam
age:19.5

 manager2
name:Sadaf
age:18

 manager3
name:Shradha
age:18
```

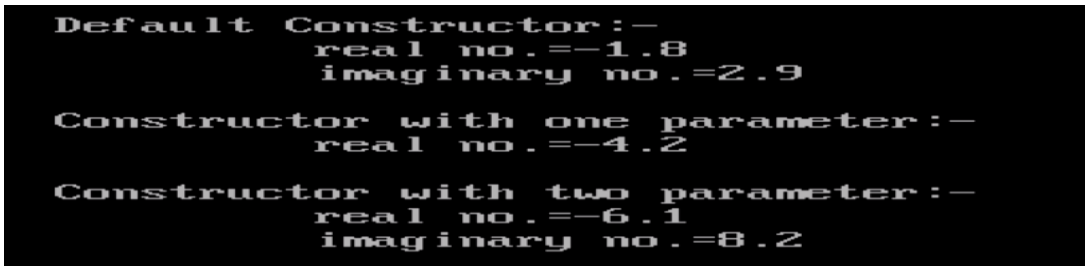
Practical 10

Create a class "complex" with real and imaginary numbers and to initialise them write overload constructor for default constructor, constructor with one parameter, constructor with two parameter.

CODE:-

```
#include<iostream.h>
#include<conio.h>
class complex
{
private:
float real;
float img;
public:
complex()
{
real=-1.8;
img=2.9;
cout<<endl<<"Default Constructor:-"<<endl;
cout<<"\t real no.="<<real<<endl;
cout<<"\t imaginary no.="<<img<<endl;
}
complex(float r)
{
real=r;
cout<<endl<<"Constructor with one parameter:-"<<endl;
cout<<"\t real no.="<<real<<endl;
}
complex (float r,float i)
{
real=r;
img=i;
cout<<endl<<"Constructor with two parameter:-"<<endl;
cout<<"\t real no.="<<real<<endl;
cout<<"\t imaginary no.="<<img<<endl;
}
};
void main()
{
clrscr();
complex c1,c2(-4.2),c3(-6.1,8.2);
getch();
}
```

OUTPUT:-



Practical 11

Create a class "employee" with empno, ename, salary as data members & create copy constructor to create objects from already created objects.

CODE:-

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
class employee
{
int empno;
char ename[30];
float salary;
public:
employee(int eno,char ena[30],float s)
{
empno=eno;
strcpy (ename,ena);
salary=s;
}
employee(employee&e)
{
empno=e.empno;
strcpy(ename,e.ename);
salary=e.salary;
}
void display()
{
cout<<"employee no.="<<empno<<endl;
cout<<"employee name="<<ename<<endl;
cout<<"employee salary="<<salary<<endl<<endl;
}
};
void main()
{
employee e1(27,"abc",20000.021);
employee e2=e1;
clrscr();
cout<<"normal parameterized constructor:"<<endl;
e1.display();
cout<<"copy constructor:"<<endl;
e2.display();
getch();
}
```

OUTPUT:-

```
normal parameterized constructor:
employee no.=27
employee name=abc
employee salary=20000.021484

copy constructor:
employee no.=27
employee name=abc
employee salary=20000.021484
```

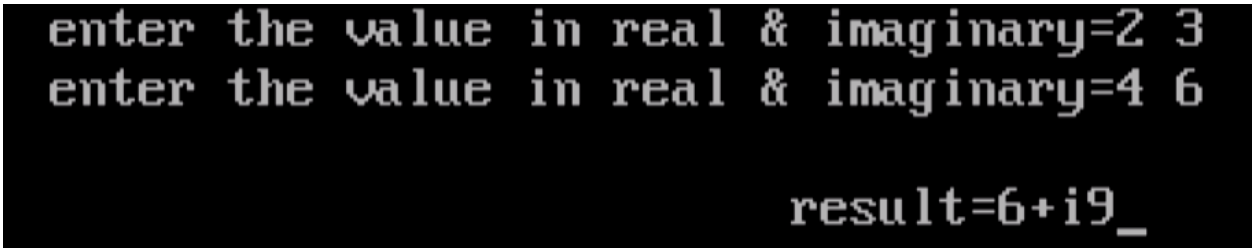
Practical 12

A program to overload binary "+" operator for complex class.

CODE:-

```
#include<iostream.h>
#include<conio.h>
class complex
{
private:
int real;
int imag;
public:
void accept()
{
cout<<"enter the value in real & imaginary=";
cin>>real>>imag;
}
complex operator+(complex q)
{
complex t;
t.real=real+q.real;
t.imag=imag+q.imag;
return(t);
}
void display()
{
cout<<"\n\t\t\t result="<<real<<"+"<<"i"<<imag;
}
};
void main()
{
complex a,b,c;
clrscr();
a.accept();
b.accept();
c=a+b;
c.display();
getch();
}
```

OUTPUT:-



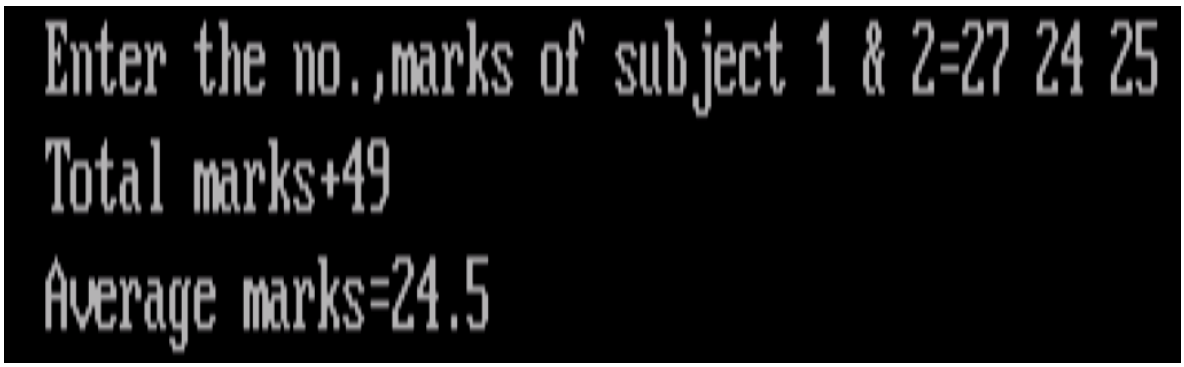
Practical 13

Write a program to single inheritance for following structure. student class (rollno., sub1, sub2) & result class (total, avg).

CODE:-

```
#include<iostream.h>
#include<conio.h>
class student
{
public:
int rollno;
float sub1, sun2;
void getdata()
{
cout<<"Enter the no.,marks of subkject 1&2=";
cin>>rollno>>sub1>>sun2;
}
};
class result:public student
{
public:
float total,avg;
void outcome(0)
{
total=sub1+sub2;
avg+(sub1+sub2)/2;
cout<<"Total marks+"<<total<<endl;
cout<<"Average marks="<<avg;
}
};
void main()
{
result r;
clrscr();
r.getdata();
r.outcome();
getch();
}
```

OUTPUT:-



Practical 14

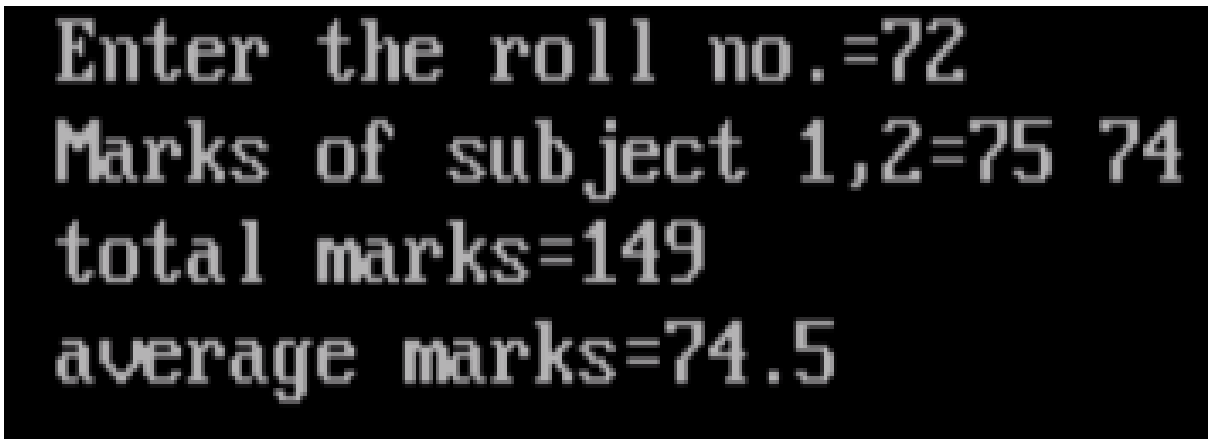
Write a class for multilevel inheritance for following structure. student class (rollno), test class (sub1, sub2) & result class (total, avg).

CODE:-

```
#include<iostream.h>
#include<conio.h>
class student
{
public:
int rollno;
void getdata()
{
cout<<"Enter the roll no.=";
cin>>rollno;
}
};
class test:public student
{
public:
float sub 1,sub2;
void indata()
{
cout<<"Marks of subject 1,2=";
cin>>sub1>>sub2;
}
};
class result:public test
{
public:
float total,avg;
void outcome()
{
total=sub1+sub2;
avg=(sub1+sub2)/2;
cout<<"total marks="<<total<<endl;
cout<,"average marks="<<avg;
]
};
void main()
{
result r;
clrscr();
r.getdata();
r.indata();
r.outcome();
grtch();
```

}

OUTPUT:-


Enter the roll no.=72
Marks of subject 1,2=75 74
total marks=149
average marks=74.5

Practical 15

C++ Program to Find the Largest Number using if-else.

CODE:-

```
// C++ Program to Find Largest Among
// Three Numbers Using if-else
// Statement
#include <bits/stdc++.h>
using namespace std;
// Driver code
int main()
{
 int a, b, c;
 cout << "Enter the three numbers a, b & c" << endl;
 cin >> a >> b >> c;
 if (a >= b) {
 // If 'a' is greater than or equal to 'b', compare
 // 'a' with 'c'
 if (a >= c) {
 // If 'a' is also greater than or equal to
'c',
 // it is the largest number
 cout << "The Largest Among Three Numbers is : "
 << a << endl;
 }
 else {
 // If 'a' is not greater than or equal to
'c',
 // 'c' must be the largest number
 cout << "The Largest Among Three Numbers is : "
 << c << endl;
 }
 }
 else {
 // If 'b' is greater than 'a', compare 'b' with 'c'
 if (b >= c) {
 // If 'b' is also greater than or equal to
'c',
 // it is the largest number
 cout << "The Largest Among Three Numbers is : "<< b << endl;
 }
 else {
 // If 'b' is not greater than or equal to
'c',
 // 'c' must be the largest number
 cout << "The Largest Among Three Numbers is : " << c << endl;
 }
 }
```

```
    return 0;
}
```

OUTPUT:-

```
Output

/tmp/fKAWfvwIdl.o
Enter the three numbers a, b & c
1 2 3
The Largest Among Three Numbers is : 3
```

Practical 16

C++ Program to Print Armstrong Numbers Between 1 to 1000

CODE:-

```
// Online C++ compiler to run C++ program online
// C++ program to find Armstrong numbers
// between 1 to 1000 using an optimized
// solution
#include <bits/stdc++.h>
using namespace std;

// Driver code
int main()
{
   int ord1, ord2, ord3, total_sum;

   cout << "All the Armstrong numbers between 1 to 1000 : ";

   // Loop which will run from 1 to 1000
   for (int num = 1; num <= 1000; ++num)
   {
      // All the single-digit numbers are
      // armstrong number.
      if (num <= 9)
      {
         cout << num << " ";
      }
      else
      {
         ord1 = num % 10;
         ord2 = (num % 100 - ord1) / 10;
         ord3 = (num % 1000 - ord2) / 100;

         total_sum = ((ord1 * ord1 * ord1) +
                  (ord2 * ord2 * ord2) +
                  (ord3 * ord3 * ord3));
         if (total_sum == num)
         {
            cout << num << " ";
         }
      }
   }
   return 0;
}
```

OUTPUT:-

```
/tmp/01NkmZv7UK.o
All the Armstrong numbers between 1 to 1000 : 1 2 3
    4 5 6 7 8 9 153 370 371 407
```