

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY
AMITY UNIVERSITY CAMPUS, SECTOR-125, NOIDA-201303



Subject- Object Oriented Programming using C++.

Subject Code- ES203

Submitted to-

Dr. Roshan Lal

Assistant Professor-III

Dept of CSE

Submitted By-

Name: Yuvansh Dua

Branch- B.Tech CSE

Section- 10Y

Enrollment No.- A2305222653

INDEX

<u>S.NO</u>	<u>Name of PROGRAM</u>	<u>PROGRAM Date</u>	<u>Evaluation Date</u>	<u>Marks Obtained</u>	<u>Remarks</u>
1.	To create multiple objects of a class.				
2.	To create class methods and how to access the class methods outside the function.				
3.	To define a class method outside the class definition.				
4.	To assign values to the private data members				
5.	To create a class to read and add two distances				
6.	To create class to get and print details of a student				
7.	To calculate the area of the wall using default constructor.				
8.	To calculate the area of the wall using parameterized constructor.				
9.	To calculate the area of the wall using copy constructor.				
10.	To calculate the volume of cube using default constructor.				

11	To calculate the volume of the cube using parameterized constructor.				
12.	To calculate the volume of cube using copy constructor.				
13.	To enter student details by giving parameters to constructors.				
14.	To demonstrate an example on simple inheritance.				
15.	To demonstrate an example of private simple inheritance.				
16.	To demonstrate an example of protected simple inheritance.				
17.	To read and print student's information using two classes and simple inheritance.				
18.	To demonstrate an example of multilevel inheritance.				
19.	To read and print employee information using multiple inheritance.				
20.	To demonstrate an example of multiple inheritance.				

21.	To demonstrate an example of hierarchical inheritance to get square and cube of a number.				
22.	To read and print employee information with department and pf information using hierarchical inheritance.				
23.	To check number palindrome and string palindrome.				
24.	To show the effect of call by value and call by reference in functions.				
25.	To perform following operations on matrix using functions and switch case: Addition, subtraction, multiplication, transpose				
26.	To Define a class Shape whose attributes are radius, length and width calculate the perimeter of the rectangle and circle using constructors and destructors.				
27.	To Create a class Person which includes character array name of size 64, age in numeric, character array address of size 64, and total salary in real numbers				

	<p>(divide salary in different components, if required) and make an array of objects of class Person of size 10, Also,</p> <p>{a)Write inline function that obtains the youngest and eldest age of a person from a class person using arrays.</p> <p>(b)Write a program to develop the salary slip and display result by using constructors.</p>				
28.	<p>Create a class called Complex for performing following operations:</p> <p>(a)Overload increment and decrement operators for increasing and decreasing complex number values (Unary operator overload).</p> <p>(b)Overload '+' op and '-' op for complex numbers (Binary operator overloading).</p>				
29.	<p>To find the area (function name AREA) of circle, rectangle and triangle by Function overloading concept.</p>				

30.	<p>Design three classes: Student, Exam and Result. The student class has data members such as roll no, name etc. Create a class Exam by inheriting the Student class. The Exam class adds data members representing the marks scored in six subjects. Derive the Result from class Exam and it has its own members such as total marks. Write an interactive program to model this relationship</p>				
31.	<p>To swap two numbers (create two classes) by using Friend function.</p>				
32.	<p>Consider an example of book shop which sells books and video tapes. These two classes are inherited from base class called media. The media class has command data members such as title and publication. The book class has data members for storing number of pages in a book and tape class has playing time in a tape. Each class will have member functions such as</p>				

	<p>read () and show ().</p> <p>In the base class, these members have to be defined as virtual functions. Write a program to models the class hierarchy for book shop and processes objects of these classes using pointers to base class. Write the rules of virtual functions.</p>				
33.	To calculate the total mark of a student using the concept of virtual base class.				
34.	To show the use of this pointer. Write the application of this pointer.				
35.	To implement stack functions using templates.				
36.	To demonstrate exception handling				
37.	<p>To input a file, which determines its length. Also count the number of word occurrence. For example:” that person is going to town to meet other person”. Here “to” and “person”- 2times.</p>				

Program-1

Objective-WAP to create multiple objects in a class.

Input-

```
#include<iostream>

#include<string>

using namespace std;

class student
{
    public:
    int rollno;
    string name;
};

int main()
{
    student std1, std2;
    std1.rollno=653;
    std1.name="Yuvansh";
    std2.rollno=657;
    std2.name="nikhil";
    cout<<"Name of Student 1:"<<std1.name<<"\n";
    cout<<"Roll No. of Student 1:"<<std1.rollno<<"\n";
    cout<<"Name of Student 2:"<<std2.name<<"\n";
    cout<<"Roll No. of Student 2:"<<std2.rollno<<"\n";
    return 0;
}
```

Output

```
Name of Student 1:Yuvansh
Roll No. of Student 1:653
Name of Student 2:nikhil
Roll No. of Student 2:657
```


Program-2

Objective-WAP to create class methods and how to access the class methods outside the function.

Input-

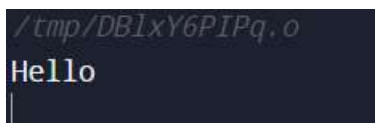
```
#include<iostream>

using namespace std;

class myclass
{
    public:
    void mymethod()
    {
        cout<<"Hello"<<endl;
    }
};

int main()
{
    myclass obj;
    obj.mymethod();
    return 0;
}
```

Output

A screenshot of a terminal window with a dark background. The top line shows a file path: /tmp/DB1xY6PIPq.o. The second line shows the output 'Hello' in a light green color. A cursor is visible at the end of the line.

```
/tmp/DB1xY6PIPq.o
Hello
|
```

Program-3

Objective- WAP to define a class method outside the class definition.

Input-

```
#include<iostream>

using namespace std;

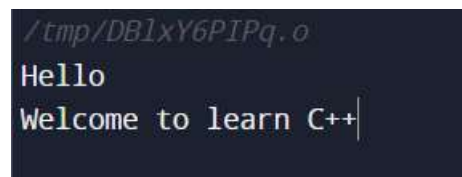
class myclass
{
    public:
    void mymethod();
    void mymethod1();
};

void myclass::mymethod()
{
    cout<<"Hello"<<endl;
}

void myclass::mymethod1()
{
    cout<<"Welcome to learn C++";
}

int main()
{
    myclass obj;
    obj.mymethod();
    obj.mymethod1();
    return 0;
}
```

Output

A screenshot of a terminal window showing the output of the C++ program. The first line is the file path `/tmp/DB1xY6PIPq.o`. The second line is the output `Hello`. The third line is the output `Welcome to learn C++` followed by a cursor.

```
/tmp/DB1xY6PIPq.o
Hello
Welcome to learn C++
```

Program-4

Objective- WAP to assign values to private data members.

Input-

```
#include<iostream>

using namespace std;

class circle
{
    private:
        double radius;
    public:
        double area()
        {
            return 3.14*radius*radius;
        }
};

int main()
{
    circle obj;
    obj.radius=2;
    cout<<"Area:"<<obj.area();
    return 0;
}
```

Output

```
main.cpp: In function 'int main()':
main.cpp:16:9: error: 'double circle::radius' is private within this context
   16 |     obj.radius=2;
      |         ^~~~~~
main.cpp:6:12: note: declared private here
    6 |     double radius;
      |         ^~~~~~
```

Program-5

Objective- WAP to create a class to read and add two distances.

Input-

```
#include <iostream>

using namespace std;

class addDistance {

public:

    // Method to add two numbers

    int add(int dis1, int dis2) {

        return dis1 + dis2;

    }

};

int main() {

    addDistance adder;

    int dis1, dis2;

    cout << "Enter first distance: ";

    cin >> dis1;

    cout << "Enter second distance: ";

    cin >> dis2;

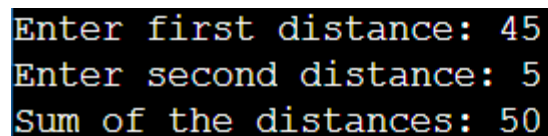
    int sum = adder.add(dis1, dis2);

    cout << "Sum of the distances: " << sum << endl;

    return 0;

}
```

Output

A screenshot of a terminal window with a black background and light blue text. It shows the output of the program: 'Enter first distance: 45', 'Enter second distance: 5', and 'Sum of the distances: 50'.

```
Enter first distance: 45
Enter second distance: 5
Sum of the distances: 50
```

Program-6

Objective-WAP to create class to get and print details of a student.

Input-

```
#include <iostream>

#include <string>

using namespace std;

class Student {

private:

    string name;

    int age;

    int rollno;

    string department;

public:

    // Method to get student details from the user

    void getDetails() {

        cout << "Enter student name: ";

        cin >> name;

        cout << "Enter student age: ";

        cin >> age;

        cout << "Enter student roll number: ";

        cin >> rollno;

        cout << "Enter student department: ";

        cin >> department;

    }

    // Method to print student details

    void printDetails() {

        cout << "Student Details:" << endl;

        cout << "Name: " << name << endl;

        cout << "Age: " << age << endl;
```

```
        cout << "Roll Number: " << rollno << endl;

        cout << "Department: " << department << endl;
    }

};

int main() {

    Student student;

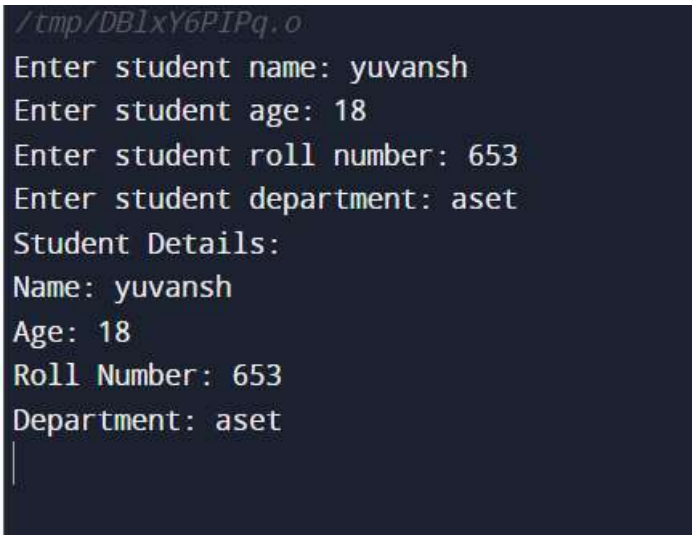
    student.getDetails();

    student.printDetails();


    return 0;

}
```

Output

A terminal window with a dark background and light-colored text. The text shows the execution of a C++ program. It starts with a file path, followed by prompts for student name, age, roll number, and department. The user enters 'yuvansh', '18', '653', and 'aset' respectively. Then, the program prints 'Student Details:' followed by the stored values: 'Name: yuvansh', 'Age: 18', 'Roll Number: 653', and 'Department: aset'. A cursor is visible at the end of the last line.

```
/tmp/DB1xY6PIPq.o
Enter student name: yuvansh
Enter student age: 18
Enter student roll number: 653
Enter student department: aset
Student Details:
Name: yuvansh
Age: 18
Roll Number: 653
Department: aset
|
```

Program-7

Objective- WAP to calculate the area of the wall using default constructor.

Input-

```
#include<iostream>

using namespace std;

class area
{
    public:
    int a,b;
    area()
    {
        a=10;
        b=20;
    }
};

int main()
{
    area c;
    cout<<"a:"<<c.a<<endl<<"b:"<<c.b<<endl;
    cout<<"Area of the wall:"<<c.a*c.b<<endl;
    return 1;
}
```

Output

```
a:10
b:20
Area of the wall:200

...Program finished with exit code 1
Press ENTER to exit console. □
```

Program-8

Objective- WAP to calculate the area of the wall using parameterized constructor.

Input-

```
#include<iostream>

using namespace std;

class area
{
    private:
    int a,b;
    public:
    area(int x, int y)
    {
        a=x;
        b=y;
    }
    int getx()
    {
        return a;
    }
    int gety()
    {
        return b;
    }
};

int main()
{
    area p(10,20);
    cout<<"a:"<<p.getx()<<endl<<"b:"<<p.gety()<<endl;
    cout<<"Area of the wall:"<<p.getx()*p.gety()<<endl;
    return 0;
}
```


Output

```
a:10  
b:20  
Area of the wall:200  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Program-9

Objective- WAP to calculate the area of the wall using copy constructor.

Input-

```
#include<iostream>

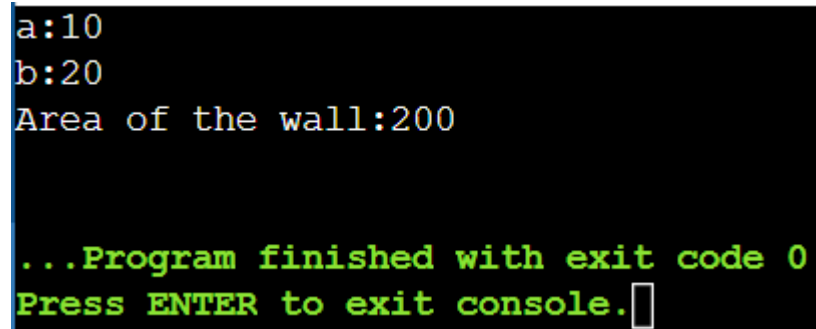
using namespace std;

class area
{
    private:
    int a,b;
    public:
    area(int x, int y)
    {
        a=x;
        b=y;
    }
    area(const area &p)
    {
        a=p.a;
        b=p.b;
    }
    int getx()
    {
        return a;
    }
    int gety()
    {
        return b;
    }
};

int main()
{
    area p1(10,20);
    area p=p1;
    cout<<"a:"<<p1.getx()<<endl<<"b:"<<p1.gety()<<endl;
```

```
cout<<"Area of the wall:"<<p.getx()*p.gety()<<endl;  
return 0;  
}
```

Output

A screenshot of a terminal window with a black background. The output of the program is displayed in a monospaced font. The first three lines are in white: 'a:10', 'b:20', and 'Area of the wall:200'. The last two lines are in green: '...Program finished with exit code 0' and 'Press ENTER to exit console.' followed by a white cursor box.

```
a:10  
b:20  
Area of the wall:200  
  
...Program finished with exit code 0  
Press ENTER to exit console.█
```

Program-10

Objective- WAP to calculate the volume of cube using default constructor.

Input-

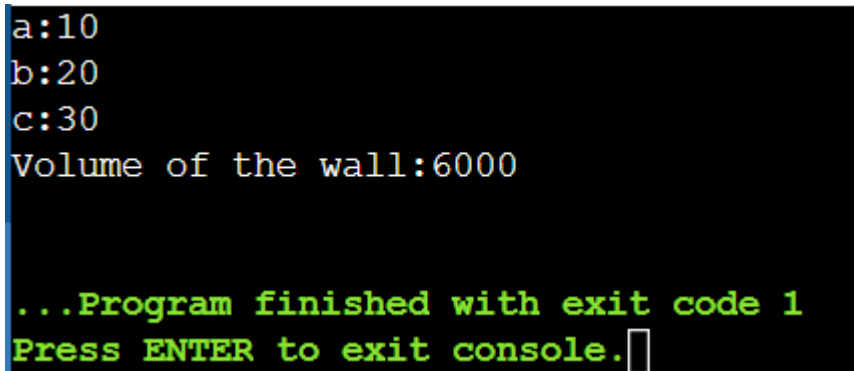
```
#include<iostream>

using namespace std;

class volume
{
    public:
    int a,b,c;
    volume()
    {
        a=10;
        b=20;
        c=30;
    }
};

int main()
{
    volume c;
    cout<<"a:"<<c.a<<endl<<"b:"<<c.b<<endl<<"c:"<<c.c<<endl;
    cout<<"Volume of the wall:"<<c.a*c.b*c.c<<endl;
    return 1;
}
```

Output



```
a:10
b:20
c:30
Volume of the wall:6000

...Program finished with exit code 1
Press ENTER to exit console. □
```

Program-11

Objective- WAP to calculate the volume of the cube using parameterized constructor.

Input-

```
#include<iostream>

using namespace std;

class volume
{
    private:
    int a,b,c;
    public:
    volume(int x, int y, int z)
    {
        a=x;
        b=y;
        c=z;
    }
    int getx()
    {
        return a;
    }
    int gety()
    {
        return b;
    }
    int getz()
    {
        return c;
    }
};

int main()
{
    volume p(10,20,30);

    cout<<"a:"<<p.getx()<<endl<<"b:"<<p.gety()<<endl<<"c:"<<p.getz()<<endl;

    cout<<"Volume of the wall:"<<p.getx()*p.gety()*p.getz()<<endl;
```

```
return 0;
```

```
}
```

Output

```
a:10  
b:20  
c:30  
Volume of the wall:6000  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Program-12

Objective- WAP to calculate the volume of cube using copy constructor.

Input-

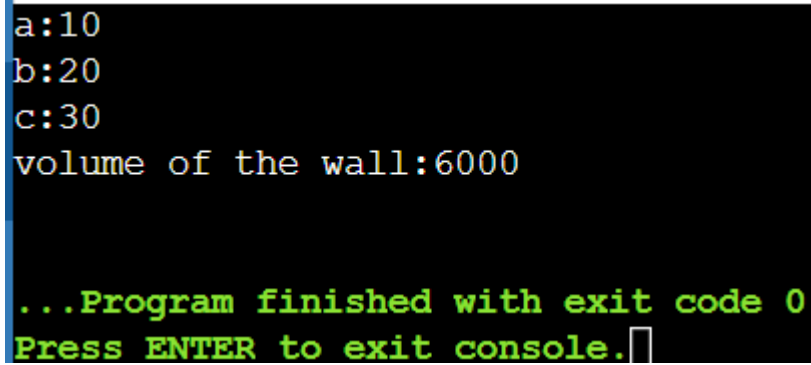
```
#include<iostream>

using namespace std;

class volume
{
    private:
    int a,b,c;
    public:
    volume(int x, int y,int z)
    {
        a=x;
        b=y;
        c=z;
    }
    volume(const volume &p)
    {
        a=p.a;
        b=p.b;
        c=p.c;
    }
    int getx()
    {
        return a;
    }
    int gety()
    {
        return b;
    }
    int getz()
    {
        return c;
    }
}
```

```
};  
  
int main()  
{  
    volume p1(10,20,30);  
    volume p=p1;  
    cout<<"a:"<<p1.getx()<<endl<<"b:"<<p1.gety()<<endl<<"c:"<<p1.getz()<<endl;;  
    cout<<"volume of the wall:"<<p.getx()*p.gety()*p.getz()<<endl;  
    return 0;  
}
```

Output



```
a:10  
b:20  
c:30  
volume of the wall:6000  
  
...Program finished with exit code 0  
Press ENTER to exit console. █
```


Program-13

Objective- WAP to enter student details by giving parameters to constructors.

Input-

```
#include<iostream>

#include<string>

using namespace std;

class student
{
    public:
    string name;
    int rollno;
    student(string n, int r)
    {
        name=n;
        rollno=r;
    }
    void display()
    {
        cout<<"Name of the student:"<<name<<endl;
        cout<<"Roll No. of the student:"<<rollno<<endl;
    }
};

int main()
{
    int r;
    string n;
    cout<<"Enter Student's name:";
    cin>>n;
    cout<<"Enter Student's Roll No.:";
    cin>>r;
    student s(n,r);
    cout<<"Student's Details"<<endl;
    s.display();
    return 0;}
```

Output

```
/tmp/DB1xY6PIPq.o
Enter Student's name:yuvansh
Enter Student's Roll No.:653
Student's Details
Name of the student:yuvansh
Roll No. of the student:653
```

Program-14

Objective- WAP to demonstrate an example on simple inheritance.

Input-

```
#include<iostream>

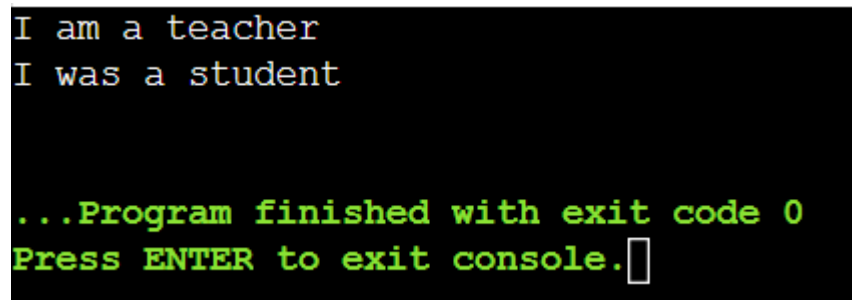
using namespace std;

class teacher
{
    public:
    teacher() {
        cout<<"I am a teacher"<<endl;
    }
};

class student:public teacher
{
    public:
    student() {
        cout<<"I was a student"<<endl;
    }
};

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

Output

A screenshot of a console window with a black background. The output text is displayed in a monospaced font. The first two lines are "I am a teacher" and "I was a student" in white. The last two lines, "...Program finished with exit code 0" and "Press ENTER to exit console.", are in green. A white cursor is visible at the end of the last line.

```
I am a teacher
I was a student

...Program finished with exit code 0
Press ENTER to exit console.
```

Program-15

Objective- WAP to demonstrate an example of private simple inheritance.

Input-

```
#include<iostream>

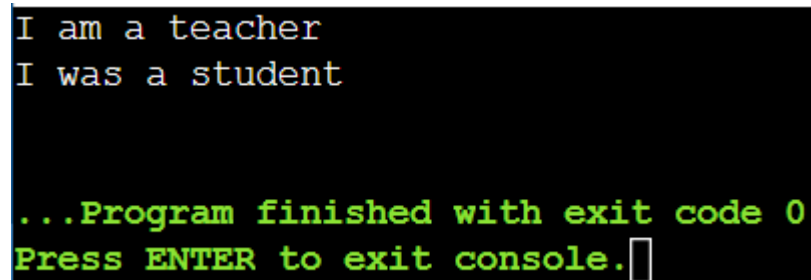
using namespace std;

class teacher
{
    public:
    teacher() {
        cout<<"I am a teacher"<<endl;
    }
};

class student:private teacher
{
    public:
    student() {
        cout<<"I was a student"<<endl;
    }
};

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

Output

A screenshot of a console window with a black background. The first two lines of output are "I am a teacher" and "I was a student" in a light blue monospaced font. The third line, "...Program finished with exit code 0", and the fourth line, "Press ENTER to exit console.", are in a bright green monospaced font. A white cursor is visible at the end of the fourth line.

```
I am a teacher
I was a student

...Program finished with exit code 0
Press ENTER to exit console.
```

Program-16

Objective- WAP to demonstrate an example of protected simple inheritance.

Input-

```
#include<iostream>

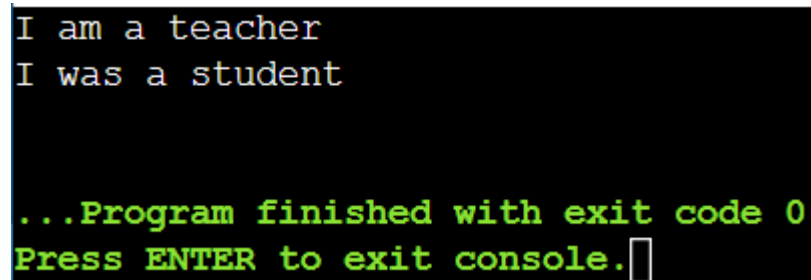
using namespace std;

class teacher
{
    public:
    teacher() {
        cout<<"I am a teacher"<<endl;
    }
};

class student:protected teacher
{
    public:
    student() {
        cout<<"I was a student"<<endl;
    }
};

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

Output

A screenshot of a terminal window with a black background. The first two lines of output are "I am a teacher" and "I was a student" in a light blue monospaced font. The third line, "...Program finished with exit code 0", and the fourth line, "Press ENTER to exit console.", are in a bright green monospaced font. A white cursor is visible at the end of the fourth line.

```
I am a teacher
I was a student

...Program finished with exit code 0
Press ENTER to exit console.
```

Program-17

Objective- WAP to read and print student's information using two classes and simple inheritance.

Input-

```
#include <iostream>

using namespace std;

class std_basic_info {
private:
    char name[30];
    int age;
    char gender;

public:
    void getBasicInfo(void);
    void putBasicInfo(void);
};

void std_basic_info::getBasicInfo(void)
{
    cout << "Enter student's basic information:" << endl;
    cout << "Name: ";
    cin >> name;
    cout << "Age: ";
    cin >> age;
    cout << "Gender: ";
    cin >> gender;
}

void std_basic_info::putBasicInfo(void) {
    cout << "Information--" << endl << "Name: " << name << endl << "Age: " << age << endl << "Gender: " <<
gender << endl;
}

class std_result_info : public std_basic_info {
private:
    int totalM;
    float perc;
    char grade;

public:
```

```

    void getResultInfo(void);

    void putResultInfo(void);

};

void std_result_info::getResultInfo(void)
{
    cout << "Enter student's result information:" << endl;

    cout << "Total Marks: ";

    cin >> totalM;

    perc = (float)((totalM * 100) / 500);

    cout << "Grade: ";

    cin >> grade;

}

void std_result_info::putResultInfo(void)
{
    cout << "Total Marks: " << totalM << endl << "Percentage: " << perc << "%" << endl << "Grade: " << grade << endl;

}

int main()
{
    std_result_info std;

    std.getBasicInfo();

    std.getResultInfo();

    std.putBasicInfo();

    std.putResultInfo();

    return 0;

}

```

Output

```
/tmp/DBLxY6PIPq.o
```

```
Enter student's basic information:
```

```
Name: yuvansh
```

```
Age: 18
```

```
Gender: male
```

```
Enter student's result information:
```

```
Total Marks: Grade: Information--
```


```
Name: yuvansh
```

```
Age: 18
```

```
Gender: m
```

```
Total Marks: 0
```

```
Percentage: 0%
```

```
Grade: 
```


Program-18

Objective- WAP to demonstrate an example of multilevel inheritance.

Input-

```
#include <iostream>

using namespace std;

class vehicle
{
    public:
    vehicle()
    {
        cout<<"This is a vehicle"<<endl;
    }
};

class fourwheeler:public vehicle
{
    public:
    fourwheeler()
    {
        cout<<"Objects with 4 wheels are vehicles"<<endl;
    }
};

class car:public fourwheeler
{
    public:
    car()
    {
        cout<<"Car has 4 wheels"<<endl;
    }
};

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

Output

```
This is a vehicle  
Objects with 4 wheels are vehicles  
Car has 4 wheels  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Program-19

Objective- WAP to read and print employee information using multiple inheritance.

Input-

```
#include <iostream>

using namespace std;

class basicInfo {
protected:
    char name[30];
    int empId;
public:
    void getBasicInfo(void)
    {
        cout << "Enter Name: ";
        cin.getline(name, 30);
        cout << "Enter Emp. Id: ";
        cin >> empId;
    }
};

class deptInfo {
protected:
    char deptName[30];
    char designation[30];
public:
    void getDeptInfo(void)
    {
        cout << "Enter Department Name: ";
        cin.ignore(1);
        cin.getline(deptName, 30);
        cout << "Designation: ";
        cin.getline(designation, 30);
    }
};

class employee : private basicInfo, private deptInfo {
public:
```

```

void getEmployeeInfo(void)
{
    cout << "Enter employee's basic info: " << endl;
    getBasicInfo();
    cout << "Enter employee's department info: " << endl;
    getDeptInfo();
}

void printEmployeeInfo(void)
{
    cout << "Employee's Information is: " << endl;
    cout << "Basic Information...:" << endl;
    cout << "Name: " << name << endl;
    cout << "Employee ID: " << empId << endl;
    cout << "Department Information...:" << endl;
    cout << "Department Name: " << deptName << endl;
    cout << "Designation: " << designation << endl;
}

};

int main()
{
    employee emp;
    emp.getEmployeeInfo();
    emp.printEmployeeInfo();
    return 0;}

```

Output

```

Enter employee's basic info:
Enter Name: yuvansh
Enter Emp. Id: 653
Enter employee's department info:
Enter Department Name: aset
Designation: manager
Employee's Information is:
Basic Information...:
Name: yuvansh
Employee ID: 653
Department Information...:
Department Name: aset
Designation: manager

```

Program-20

Objective- WAP to demonstrate an example of multiple inheritance.

Input-

```
#include <iostream>

using namespace std;

class vehicle
{
    public:
    vehicle()
    {
        cout<<"This is a vehicle"<<endl;
    }
};

class fourwheeler
{
    public:
    fourwheeler()
    {
        cout<<"Objects with 4 wheels are vehicles"<<endl;
    }
};

class car:public vehicle, public fourwheeler
{
    public:
    car()
    {
        cout<<"Car has 4 wheels"<<endl;
    }
};

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

Output

```
This is a vehicle  
Objects with 4 wheels are vehicles  
Car has 4 wheels  
  
...Program finished with exit code 0  
Press ENTER to exit console.[]
```

Program-21

Objective- WAP to demonstrate an example of hierarchical inheritance to get square and cube of a number.

Input-

```
#include <iostream>

using namespace std;

class number
{
    private:
    int num;
    public:
    number()
    {
        cout<<"Enter any number:";
        cin>>num;

    }

    int returnnum()
    {
        return num;
    }
};

class square: public number
{
    public:
    square()
    {
        int num;
        num=returnnum();
        cout<<"The sqaure of the number is:"<<num*num<<endl;
    }
};

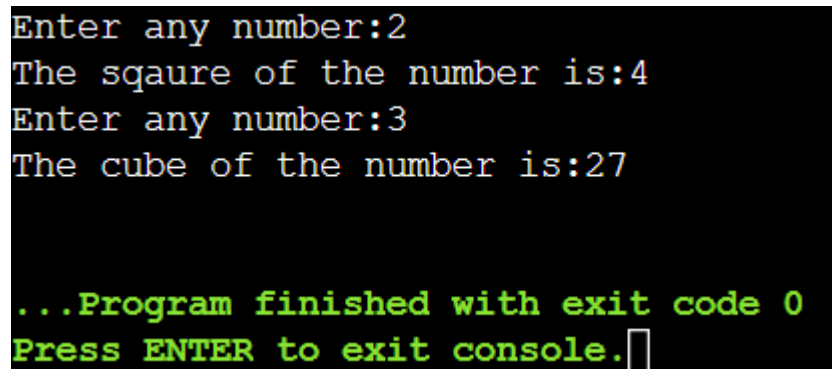
class cube:public number
{

```

```
public:
cube()
{
    int num;
    num=returnnum();
    cout<<"The cube of the number is:"<<num*num*num<<endl;
}
};

int main()
{
    square obj1;
    cube obj2;
    return 0;
}
```

Output



```
Enter any number:2
The sqaure of the number is:4
Enter any number:3
The cube of the number is:27

...Program finished with exit code 0
Press ENTER to exit console.
```


Program-22

Objective- WAP to read and print employee information with department and pf information using hierarchical inheritance.

Input-

```
#include <iostream>

using namespace std;

class basicInfo {

protected:

    char name[30];

    int empId;

public:

    void getBasicInfo(void)

    {

        cout << "Enter Name: ";

        cin.getline(name, 30);

        cout << "Enter Emp. Id: ";

        cin >> empId;

    }

};

class deptInfo : private basicInfo {

protected:

    char deptName[30];

    char designation[30];

public:

    void getDeptInfo(void)

    {

        getBasicInfo();

        cout << "Enter Department Name: ";

        cin.ignore(1);

        cin.getline(deptName, 30);

        cout << "Enter Designation: ";

        cin.getline(designation, 30);

    }

    void printDeptInfo(void)
```

```

{
    cout << "Employee's Information is: " << endl;
    cout << "Basic Information...:" << endl;
    cout << "Name: " << name << endl;
    cout << "Employee ID: " << empId << endl;

    cout << "Department Information...:" << endl;
    cout << "Department Name: " << deptName << endl;
    cout << "Designation: " << designation << endl;

}
};

class loanInfo : private basicInfo {
protected:
    char loanDetails[30];
    int loanAmount;

public:
    void getLoanInfo(void)
    {
        getBasicInfo();
        cout << "Enter Loan Details: ";
        cin.ignore(1);
        cin.getline(loanDetails, 30);
        cout << "Enter loan amount: ";
        cin >> loanAmount;
    }

    void printLoanInfo(void)
    {
        cout << "Employee's Information is: " << endl;
        cout << "Basic Information...:" << endl;
        cout << "Name: " << name << endl;
        cout << "Employee ID: " << empId << endl;
    }
};

```

```
        cout << "Loan Information...:" << endl;
        cout << "Loan Details: " << loanDetails << endl;
        cout << "Loan Amount : " << loanAmount << endl;
    }
};
```

```
int main()
{
    deptInfo objD;

    objD.getDeptInfo();
    objD.printDeptInfo();

    cout << endl
        << endl;
    loanInfo objL;
    objL.getLoanInfo();
    objL.printLoanInfo();

    return 0;
}
```

Output

```
/tmp/DB1xY6PIPq.o
Enter Name: yuvansh
Enter Emp. Id: 653
Enter Department Name: aset
Enter Designation: manager
Employee's Information is:
Basic Information...:
Name: yuvansh
Employee ID: 653
Department Information...:
Department Name: aset
Designation: manager

Enter Name: kanak
Enter Emp. Id: 644
Enter Loan Details: 2000000
Enter loan amount: 20000000
Employee's Information is:
Basic Information...:
Name: kanak
Employee ID: 644
Loan Information...:
Loan Details: 2000000
Loan Amount : 20000000
```

Program-23

Objective- WAP To check number palindrome and string palindrome.

Input-

```
#include <iostream>
```

```
#include <string>
```

```
#include <algorithm>
```

```
bool isPalindromeNumber(int number) {
```

```
    int originalNumber = number;
```

```
    int reversedNumber = 0;
```

```
    while (number > 0) {
```

```
        int digit = number % 10;
```

```
        reversedNumber = reversedNumber * 10 + digit;
```

```
        number /= 10;
```

```
    }
```

```
    return originalNumber == reversedNumber;
```

```
}
```

```
bool isPalindromeString(const std::string& str) {
```

```
    std::string strippedStr;
```

```
    std::remove_copy_if(str.begin(), str.end(), std::back_inserter(strippedStr), [](char c) {
```

```
        return std::isspace(c);
```

```
    });
```

```
    std::transform(strippedStr.begin(), strippedStr.end(), strippedStr.begin(), ::tolower);
```

```
    std::string reversedStr(strippedStr.rbegin(), strippedStr.rend());
```

```
    return strippedStr == reversedStr;
```

```
}
```

```
int main() {
```

```
    std::cout << "Enter a number or string: ";
```

```

std::string input;
std::getline(std::cin, input);

if (std::all_of(input.begin(), input.end(), ::isdigit)) {
    int number = std::stoi(input);
    if (isPalindromeNumber(number)) {
        std::cout << number << " is a palindrome as a number." << std::endl;
    } else {
        std::cout << number << " is not a palindrome as a number." << std::endl;
    }
} else {
    if (isPalindromeString(input)) {
        std::cout << "'" << input << "' is a palindrome as a string." << std::endl;
    } else {
        std::cout << "'" << input << "' is not a palindrome as a string." << std::endl;
    }
}

return 0;
}

```

Output-

```

Enter a number or string: 232
232 is a palindrome as a number.
|

```

```

Enter a number or string: hello
'hello' is not a palindrome as a string.
|

```

```

Enter a number or string: oooo
'oooo' is a palindrome as a string.
|

```

Program-24

Objective- WAP To show the effect of call by value and call by reference in functions.

Input-

```
#include <iostream>

void incrementByValue(int x) {
    x++;
    std::cout << "Inside incrementByValue function: x = " << x << std::endl;
}

void incrementByReference(int &x) {
    x++;
    std::cout << "Inside incrementByReference function: x = " << x << std::endl;
}

int main() {
    int num = 5;

    std::cout << "Original value of num: " << num << std::endl;

    incrementByValue(num);
    std::cout << "After call by value: num = " << num << std::endl;

    incrementByReference(num);
    std::cout << "After call by reference: num = " << num << std::endl;

    return 0;
}
```

Output-

```
Original value of num: 5
Inside incrementByValue function: x = 6
After call by value: num = 5
Inside incrementByReference function: x = 6
After call by reference: num = 6
```

Program-25

Objective- WAP To perform following operations on matrix using functions and switch case:

Addition, subtraction, multiplication, transpose

Input-

```
#include <iostream>
```

```
using namespace std;
```

```
class p26 {
```

```
public:
```

```
void add(int x[3][3], int y[3][3], int r, int c) {
```

```
    int i, j;
```

```
    int sum[r][c];
```

```
    for(i = 0; i < r; ++i)
```

```
        for(j = 0; j < c; ++j)
```

```
            sum[i][j] = x[i][j] + y[i][j];
```

```
    for(i = 0; i < r; ++i)
```

```
        for(j = 0; j < c; ++j) {
```

```
            cout << sum[i][j] << " ";
```

```
            if(j == c - 1)
```

```
                cout << endl;
```

```
        }
```

```
    }
```

```
void subtract(int x[3][3], int y[3][3], int r, int c) {
```

```
    int i, j;
```

```
    int diff[r][c];
```

```
    for(i = 0; i < r; ++i)
```

```
        for(j = 0; j < c; ++j)
```

```
            diff[i][j] = x[i][j] - y[i][j];
```

```
    for(i = 0; i < r; ++i)
```

```
        for(j = 0; j < c; ++j) {
```

```
            cout << diff[i][j] << " ";
```

```
            if(j == c - 1)
```

```
                cout << endl;
```



```
    }  
}
```

```
void multiply(int x[3][3], int y[3][3], int r, int c) {  
    int result[r][c];  
    for (int i = 0; i < r; i++) {  
        for (int j = 0; j < c; j++) {  
            result[i][j] = 0;  
            for (int k = 0; k < r; k++) {  
                result[i][j] += x[i][k] * y[k][j];  
            }  
            cout << result[i][j] << "\\t";  
        }  
        cout << endl;  
    }  
}
```

```
void transpose(int x[3][3], int c) {  
    int i, j;  
    int y[3][3];  
    for (i = 0; i < c; i++)  
        for (j = 0; j < c; j++)  
            y[i][j] = x[j][i];  
    for (i = 0; i < c; i++){  
        for (j = 0; j < c; j++)  
            cout << y[i][j] << "\\t";  
        cout << endl;  
    }  
}  
};
```

```
int main() {  
    p26 obj1;  
    int c;  
    int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
```

```
int b[3][3] = {{11,12,13},{14,15,16},{17,18,19}};
```

```
cout << "Enter choice: (1) Addition (2) Subtraction (3) Multiplication (4) Transpose" << endl;
```

```
cin >> c;
```

```
switch(c) {
```

```
    case 1:
```

```
        obj1.add(a, b, 3, 3);
```

```
        break;
```

```
    case 2:
```

```
        obj1.subtract(a, b, 3, 3);
```

```
        break;
```

```
    case 3:
```

```
        obj1.multiply(a, b, 3, 3);
```

```
        break;
```

```
    case 4:
```

```
        obj1.transpose(a, 3);
```

```
        break;
```

```
    default:
```

```
        cout << "Invalid choice";
```

```
}
```

```
return 0;
```

```
}
```

Output-

```
Matrix A:
1 2 3
4 5 6
7 8 9
Matrix B:
11 12 13
14 15 16
17 18 19
Choose an operation:
1. Addition
2. Subtraction
3. Multiplication
4. Transpose
5. Exit
1
12 14 16
18 20 22
24 26 28
Choose an operation:
1. Addition
2. Subtraction
3. Multiplication
4. Transpose
5. Exit
2
-10 -10 -10
-10 -10 -10
-10 -10 -10
```

```
Choose an operation:
1. Addition
2. Subtraction
3. Multiplication
4. Transpose
5. Exit
3
90 96 102
216 231 246
342 366 390
Choose an operation:
1. Addition
2. Subtraction
3. Multiplication
4. Transpose
5. Exit
4
1 4 7
2 5 8
3 6 9
Choose an operation:
1. Addition
2. Subtraction
3. Multiplication
4. Transpose
5. Exit
5
Exiting the program.
```

Program-26

Objective- WAP to Define a class Shape whose attributes are radius, length and width calculate the perimeter of the rectangle and circle using constructors and destructors.

Input-

```
#include <iostream>
```

```
#include <cmath>
```

```
class Shape {
```

```
private:
```

```
    double radius;
```

```
    double length;
```

```
    double width;
```

```
public:
```

```
    // Constructor for a circle
```

```
    Shape(double r) : radius(r), length(0.0), width(0.0) {
```

```
        std::cout << "Circle created with radius: " << radius << std::endl;
```

```
    }
```

```
    // Constructor for a rectangle
```

```
    Shape(double l, double w) : radius(0.0), length(l), width(w) {
```

```
        std::cout << "Rectangle created with length: " << length << " and width: " << width << std::endl;
```

```
    }
```

```
    // Destructor
```

```
    ~Shape() {
```

```
        std::cout << "Shape object destroyed." << std::endl;
```

```
    }
```

```
    // Calculate and display the perimeter of a circle
```

```
    double calculateCirclePerimeter() {
```

```
        double perimeter = 2 * M_PI * radius;
```

```
        return perimeter;
```

```
    }
```

```
// Calculate and display the perimeter of a rectangle

double calculateRectanglePerimeter() {
    double perimeter = 2 * (length + width);
    return perimeter;
}

};

int main() {
    // Create a circle and calculate its perimeter
    Shape circle(5.0);
    double circlePerimeter = circle.calculateCirclePerimeter();
    std::cout << "Perimeter of the circle: " << circlePerimeter << std::endl;

    // Create a rectangle and calculate its perimeter
    Shape rectangle(4.0, 3.0);
    double rectanglePerimeter = rectangle.calculateRectanglePerimeter();
    std::cout << "Perimeter of the rectangle: " << rectanglePerimeter << std::endl;

    return 0;
}
```

Output-

```
Circle created with radius: 5
Perimeter of the circle: 31.4159
Rectangle created with length: 4 and width: 3
Perimeter of the rectangle: 14
Shape object destroyed.
Shape object destroyed.
```

Program-27

Objective- WAP to Create a class Person which includes: character array name of size 64, age in numeric, character array address of size 64, and total salary in real numbers (divide salary in different components, if required) and make an array of objects of class Person of size 10, Also,

{a)Write inline function that obtains the youngest and eldest age of a person from a class person using arrays.

(b)Write a program to develop the salary slip and display result by using constructors.

Input-

```
#include <iostream>
```

```
#include <cstring>
```

```
using namespace std;
```

```
class Person {
```

```
public:
```

```
    char name[64], address[64];
```

```
    int age, salary;
```

```
    Person(char* n, char* ad, int a, int s) {
```

```
        strcpy(name, n);
```

```
        strcpy(address, ad);
```

```
        age = a;
```

```
        salary = s;
```

```
        cout << "Salary Slip:" << endl << name << endl << address << endl
```

```
            << age << endl << salary << endl;
```

```
    }
```

```
    void maxmi(int i, int* max, int* mi);
```

```
};
```

```
inline void Person::maxmi(int i, int* max, int* mi) {
```

```
    if (i < *mi)
```

```
        *mi = i;
```

```
    if (i > *max)
```

```
        *max = i;
```

```
}
```

```
int main() {  
    char name[64], ad[64];  
    int max = 0, mi = 1000, i, a, s;  
    Person* obj = (Person*)malloc(10 * sizeof(Person));  
  
    for (i = 0; i < 10; i++) {  
        cout << "Enter name, address, age, and salary" << endl;  
        cin >> name >> ad >> a >> s;  
        obj[i] = Person(name, ad, a, s);  
        obj[i].maxmi(obj[i].age, &max, &mi);  
    }  
  
    cout << "Youngest age = " << mi << endl;  
    cout << "Oldest age = " << max << endl;  
}
```

Output-

```
Enter name, address, age, and salary
yuvansh
abc
18
2000000
Salary Slip:
yuvansh
abc
18
2000000
Enter name, address, age, and salary
nikhil
def
19
200000
Salary Slip:
nikhil
def
19
200000
Enter name, address, age, and salary
hitesh
hij
20
100000
Salary Slip:
hitesh
hij
20
100000
Youngest age = 18
Oldest age = 20
```


Program-28

Objective- WAP to Create a class called Complex for performing following operations:

(a)Overload increment and decrement operators for increasing and decreasing complex number values (Unary operator overload).

(b)Overload '+' op and '-' op for complex numbers (Binary operator overloading).

Input-

```
#include <iostream>
```

```
using namespace std;
```

```
class Complex {
```

```
public:
```

```
    int r, i;
```

```
    Complex(int r1 = 0, int i1 = 0) {
```

```
        r = r1;
```

```
        i = i1;
```

```
    }
```

```
    void operator--() {
```

```
        r--;
```

```
        i--;
```

```
        cout << "Decrement: " << r << "+" << i << "i" << endl;
```

```
    }
```

```
    void operator++() {
```

```
        r++;
```

```
        i++;
```

```
        cout << "Increment: " << r << "+" << i << "i" << endl;
```

```
    }
```

```
    Complex operator+(Complex C2) {
```

```
        Complex obj1;
```

```
        obj1.r = r + C2.r;
```

```
        obj1.i = i + C2.i;
```

```
    return obj1;  
}
```

```
Complex operator-(Complex C2) {  
    Complex obj1;  
    obj1.r = r - C2.r;  
    obj1.i = i - C2.i;  
    return obj1;  
}  
};
```

```
int main() {  
    Complex C1(2, 3);  
    Complex C2(3, 4);  
    Complex C3, C4;  
  
    --C1;  
    ++C1;  
  
    C3 = C1 + C2;  
    cout << "Addition: " << C3.r << "+" << C3.i << "i" << endl;  
  
    C4 = C1 - C2;  
    cout << "Subtraction: " << C4.r << "+" << C4.i << "i" << endl;  
}
```

Output-

```
Decrement: 1+2i  
Increment: 2+3i  
Addition: 5+7i  
Subtraction: -1+-1i
```

Program-29

Objective- WAP to find the area (function name AREA) of circle, rectangle and triangle by Function overloading concept.

Input-

```
#include <cmath>

#include <iostream>

using namespace std;

class p30 {
public:
    void area(int r) {
        cout << "Area of circle = " << 3.14 * r * r << endl;
    }

    void area(int l, int b) {
        cout << "Area of rectangle = " << l * b << endl;
    }

    void area(int a, int b, int c) {
        double s = (a + b + c) / 2;
        cout << "Area of triangle = " << sqrt(s * (s - a) * (s - b) * (s - c)) << endl;
    }
};

int main() {
    p30 obj1;
    obj1.area(5);
    obj1.area(2, 4);
    obj1.area(5, 6, 7);
}
```

Output-

```
Area of circle = 78.5
Area of rectangle = 8
Area of triangle = 14.6969
```

Program-30

Objective- WAP to Design three classes: Student, Exam and Result. The student class has data members such as roll no, name etc. Create a class Exam by inheriting the Student class. The Exam class adds data members representing the marks scored in six subjects. Derive the Result from class Exam and it has its own members such as total marks. Write an interactive program to model this relationship.

Input-

```
#include <string>

#include <iostream>

using namespace std;

class Student {
public:
    int roll_no;
    string name;
};

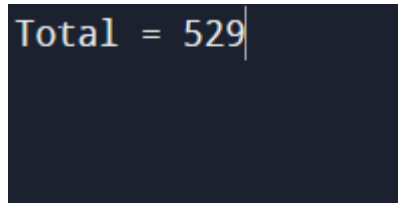
class exam : public Student {
public:
    int math, english, chem, bio, phy, cse;
};

class result : public exam {
public:
    int total;
};

int main() {
    result obj1;
    obj1.math = 89;
    obj1.english = 79;
    obj1.chem = 88;
    obj1.bio = 89;
    obj1.phy = 85;
    obj1.cse = 99;
```

```
obj1.total = obj1.math + obj1.english + obj1.chem + obj1.bio + obj1.phy + obj1.cse;  
cout << "Total = " << obj1.total;  
}
```

Output-

A dark-themed terminal window showing the output of the program. The text "Total = 529" is displayed in a light blue or cyan monospace font. A vertical cursor line is positioned at the end of the text.

```
Total = 529
```

Program-31

Objective- WAP to swap two numbers (create two classes) by using Friend function.

Input-

```
#include <iostream>

using namespace std;

class p32 {
public:
    int a = 5;
};

class p32b {
public:
    int b = 6;
    friend void swap(p32, p32b);
};

void swap(p32 obj1, p32b obj2) {
    cout << "Before swap: " << obj1.a << "\t" << obj2.b << endl;
    int t;
    t = obj1.a;
    obj1.a = obj2.b;
    obj2.b = t;
    cout << "After swap: " << obj1.a << "\t" << obj2.b << endl;
}

int main() {
    p32 obj1;
    p32b obj2;
    swap(obj1, obj2);
}
```

Output-

```
Before swap: 5  6
After swap: 6  5
```

Program-32

Objective- WAP to Consider an example of book shop which sells books and video tapes. These two classes are inherited from base class called media. The media class has command data members such as title and publication. The book class has data members for storing number of pages in a book and tape class has playing time in a tape. Each class will have member functions such as read () and show (). In the base class, these members have to be defined as virtual functions. Write a program to models the class hierarchy for book shop and processes objects of these classes using pointers to base class. Write the rules of virtual functions.

Input-

```
#include <iostream>
```

```
using namespace std;
```

```
class media {
```

```
public:
```

```
    string title, publication;
```

```
    virtual void show() {}
```

```
    virtual int read() {}
```

```
};
```

```
class books : public media {
```

```
public:
```

```
    int page;
```

```
    int read() {
```

```
        return page;
```

```
    }
```

```
    void show() {
```

```
        cout << "Number of pages = " << page << endl;
```

```
    }
```

```
};
```

```
class tapes : public media {
```

```
public:
```

```
    int time;
```

```
    int read() {
```

```
        return time;
```

```
    }  
    void show() {  
        cout << "Playing time = " << time << endl;  
    }  
};
```

```
int main() {  
    books obj1;  
    tapes obj2;  
    obj1.page = 20;  
    obj2.time = 79;  
    obj1.show();  
    obj2.show();  
}
```

Output-

```
Number of pages = 20  
Playing time = 79
```


Program-33

Objective- WAP to calculate the total mark of a student using the concept of virtual base class.

Input-

```
#include <iostream>

using namespace std;

class p34v {
public:
    virtual void calculate() = 0;
};

class p34 : public p34v {
public:
    int maths = 98;
    int cse = 90;

    void calculate() {
        cout << "Total = " << maths + cse;
    }
};

int main() {
    p34 obj1;
    obj1.calculate();
}
```

Output-

```
Total = 188
```

Program-34

Objective- WAP to show the use of this pointer. Write the application of this pointer.

Input-

```
#include <iostream>

using namespace std;

class Box {
private:
    double length;
    double width;
    double height;

public:
    // Constructor to initialize the Box object
    Box(double length, double width, double height) {
        this->length = length; // Using 'this' to access the member variable
        this->width = width;
        this->height = height;
    }

    // Function to calculate the volume of the Box
    double calculateVolume() {
        return this->length * this->width * this->height; // Using 'this' to access member variables
    }
};

int main() {
    Box myBox(5.0, 3.0, 2.0);

    // Calculate and display the volume using the calculateVolume() method
    cout << "Volume of the Box: " << myBox.calculateVolume() << " cubic units" << endl;

    return 0;
}
```

Output-

```
Volume of the Box: 30 cubic units
```

Program-35

Objective- WAP to implement stack functions using templates.

Input-

```
#include <iostream>
```

```
#include <vector>
```

```
template <typename T>
```

```
class Stack {
```

```
private:
```

```
    std::vector<T> elements;
```

```
public:
```

```
    void push(const T& value) {
```

```
        elements.push_back(value);
```

```
    }
```

```
    void pop() {
```

```
        if (!isEmpty()) {
```

```
            elements.pop_back();
```

```
        } else {
```

```
            std::cout << "Stack is empty. Cannot pop." << std::endl;
```

```
        }
```

```
    }
```

```
    T top() const {
```

```
        if (!isEmpty()) {
```

```
            return elements.back();
```

```
        } else {
```

```
            std::cout << "Stack is empty. Cannot access top." << std::endl;
```

```
            return T(); // Return a default value
```

```
        }
```

```
    }
```

```
    bool isEmpty() const {
```

```

        return elements.empty();
    }

    size_t size() const {
        return elements.size();
    }
};

int main() {
    Stack<int> intStack;
    Stack<double> doubleStack;

    intStack.push(5);
    intStack.push(10);
    intStack.push(15);

    doubleStack.push(3.14);
    doubleStack.push(2.718);

    std::cout << "Top of intStack: " << intStack.top() << std::endl;
    std::cout << "Top of doubleStack: " << doubleStack.top() << std::endl;

    intStack.pop();
    doubleStack.pop();

    std::cout << "Size of intStack: " << intStack.size() << std::endl;
    std::cout << "Size of doubleStack: " << doubleStack.size() << std::endl;

    return 0;
}

```

Output-

```

Top of intStack: 15
Top of doubleStack: 2.718
Size of intStack: 2
Size of doubleStack: 1
|

```

Program-36

Objective- WAP to demonstrate exception handling.

Input-

```
#include <iostream>

using namespace std;

int main() {
    int numerator, denominator;
    double result;
    cout << "Enter the numerator: ";
    cin >> numerator;
    cout << "Enter the denominator: ";
    cin >> denominator;

    try {
        if (denominator == 0) {
            throw "Division by zero is not allowed."; // Throwing a character string as an exception
        }

        result = static_cast<double>(numerator) / denominator;
        cout << "Result of division: " << result << endl;
    } catch (const char* error) {
        cerr << "Exception: " << error << endl;
    }

    cout << "Program continues after exception handling." << endl;

    return 0;
}
```

Output-

```
Enter the numerator: 20
Enter the denominator: 0
Exception: Division by zero is not allowed.
Program continues after exception handling.
```

Program-37

Objective- WAP to input a file, which determines its length. Also count the number of word occurrence. For example:” that person is going to town to meet other person”. Here “to” and “person”-2times.

Input-

```
#include <iostream>
```

```
#include <string>
```

```
int main() {
    std::string inputText;
    std::string wordToCount;

    std::cout << "Enter the text: ";
    std::getline(std::cin, inputText);

    std::cout << "Enter the word to count: ";
    std::cin >> wordToCount;

    int fileLength = inputText.length();
    int wordCount = 0;

    size_t pos = 0;
    while ((pos = inputText.find(wordToCount, pos)) != std::string::npos) {
        wordCount++;
        pos += wordToCount.length();
    }

    std::cout << "Text length: " << fileLength << " characters." << std::endl;
    std::cout << "Number of occurrences of '" << wordToCount << "': " << wordCount << std::endl;
    return 0;
}
```

Output-

```
Enter the text: hello this the code by yuvansh of cse10y
Enter the word to count: hello
Text length: 40 characters.
Number of occurrences of 'hello': 1
```