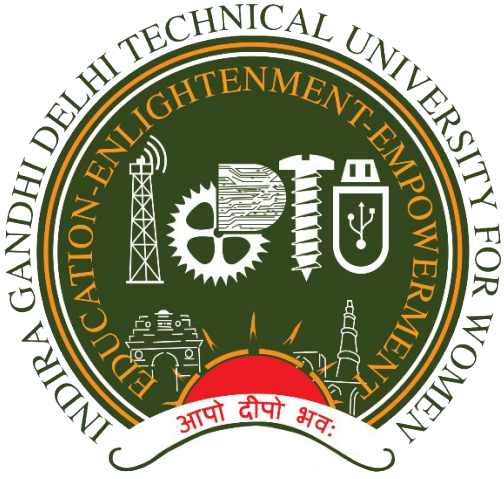
INDIRA GANDHI DELHI TECHNICAL UNIVERSITY



Object Oriented Programming (OOP)

BIT-204

SUBMITTED TO:

DR. JYOTI SHOKEEN

Assistant Professor

SUBMITTED BY:

Akshita Singh

00801182023

ECE-AI-I

INDEX

INDEX

S.NO.	Description	DATE
1.	A C++ program to find whether a given year is a leap year or not.	
2.	A C++ program to accept a coordinate point in in x-y coordinate system and determine in which quadrant the coordinate point lies in.	
3.	A C++ program to read roll no., name, marks of 3 subjects and calculate total percentage and division with grade.	
4.	A C++ program to display the N terms of odd natural no. and their sum.	
5.	A C++ program to print a set of prime numbers from 1 to 30 using continue.	
6.	Write a C++ program to perform function overloading.	
7.	Design a single C++ program to illustrate following concept of inheritance: - public derivation - private derivation - protected derivation	

8. Write a program in C++ to use scope resolution operator.

9. Write a program in C++ to print right angled triangle pyramid with '*'.

10. Write a C++ program to print an inverted right angled triangle pyramid with '*'.

11. Write a C++ program to print a right-angled triangle pyramid using numbers.

12. Design a basic calculator with basic mathematical operations using switch statement in C++.

13. Check whether a given number is even or odd using ‘goto’ in C++.

14. Write a program in C++to illustrate the use of:

- inline function

- function with default arguments

15. Write a program in C++ to read an employee's info from the user and print the same. Employee info includes employee ID using int, employee name using string, employee salary using class and object.

16. Write a program in C++ to illustrate the use of friend function.

17. Use friend function to compare the speed of car object and truck object. Declare the function as friend of both classes in C++.

18. Write a program in C++ to swap two numbers using- a) Call by value b) Call by reference c) Call by reference without pointer.

19. Write a program in C++ to demonstrate operator overloading using the '+' operator to add two complex numbers.

20. Write a program in C++ to demonstrate both static and non-static members within a class.

21. Write a program in C++ to demonstrate the concept of virtual function.

22. Write a program to illustrate the use of try, catch and throw blocks (exception handling) in C++.

23. Write a basic program in java to implement class and object.

24. Write a program in java to demonstrate the use of 'instanceof' keyword.

25. Write a program in java to demonstrate the 3 uses of super keyword:

- to access parent

class instance

variables.

- to invoke parent

class method.

- to invoke parent

class constructor.

26. Write a java program to illustrate the use of abstract class.

27. Write a java program to illustrate the use of 'interface'.

28. Write a Java program that demonstrates the use of the final keyword.

29. Write a Java program to implement Multithreading Using the Runnable Interface.

30. Write a Java program that demonstrates the use of exception handling mechanisms such as try, catch, finally, and throw to handle runtime errors effectively.

LAB EXPERIMENT- 1

Q1. Write a C++ program to find whether a given year is a leap year or not.

Ans-

```cpp
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;

int main() {
    int a;
    cout<<"Enter a year:";
    cin>>a;
    if ((a%4==0 && a%100!=0) || ( a%400==0)){
    cout<<"Leap year"<<endl;

    }
    else {
    cout<<"Not a leap year"<<endl;
    }

    return 0;
}
```

Output

```
Enter a year:2058
Not a leap year


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

Q2. To accept a coordinate point in in xy coordinate system and determine in which quadrant the coordinate point lies in.

main.cpp

```cpp
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;
int main() {
    int a,b;
    cout<<"Enter x coordinate:";
    cin>>a;
    cout<<"Enter y coordinate:";
    cin>>b;
    if (a==0 && b==0) {
    cout<<"Origin"<<endl;}
    else if (a>0 && b>0){
        cout<<"I QUADRANT";}
     else if (a<0 && b>0){
        cout<<"II QUADRANT";}

    else if (a<0 && b<0){
        cout<<"III QUADRANT";
    }
    else if (a>0 && b<0){
        cout<<"IV QUADRANT";
    }
    return 0;
}
```

Output

```
Enter x coordinate:-5
Enter y coordinate:2
II QUADRANT

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

Q3.To read roll no., name, marks of 3 subjects and calculate total percentage and division with grade.

main.cpp

```cpp
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;
int main() {
    int r, a, b, c, total;
    float percentage;
    char name[20];
    cout << "Enter roll no: ";
    cin >> r;
   cout << "Enter name: ";
    cin >> name;
  cout << "Enter sub1 marks: ";
    cin >> a;
  cout << "Enter sub2 marks: ";
    cin >> b;
cout << "Enter sub3 marks: ";
    cin >> c;
  total = a + b + c;
    percentage = (total / 300.0) * 100;
  cout << "\n--- STUDENT REPORT ---\n";
    cout << "Roll No: " << r << endl;
    cout << "Name: " << name << endl;
    cout << "Total Marks: " << total << "/300" << endl;
    cout << "Percentage: " << percentage << "%" << endl;
 if (percentage >= 60) {
        cout << "Division: First\nGrade: B" << endl;
    } else if (percentage >= 50) {
        cout << "Division: Second\nGrade: C" << endl;
    } else if (percentage >= 40) {
        cout << "Division: Third\nGrade: D" << endl;
    } else {
        cout << "Division: Fail\nGrade: F" << endl;
    }
return 0;
```

Output

```
Enter roll no: 08
Enter name: alia
Enter sub1 marks: 77
Enter sub2 marks: 89
Enter sub3 marks: 90

--- STUDENT REPORT ---
Roll No: 8
Name: alia
Total Marks: 256/300
Percentage: 85.3333%
Division: First
Grade: B
```

Q4. Display the N terms of odd natural numbers and their sum.

main.cpp

```cpp
#include <iostream>
using namespace std;
int main() {
    int n, i, sum = 0;
    cout << "Enter how many odd numbers: ";
    cin >> n;
for (i = 1; i <= 2 * n; i += 2) {
        cout << i << " ";
        sum += i;
    }

    cout << "\nSum = " << sum;
    return 0;
}
```

Output

```
Enter how many odd numbers: 10
1 3 5 7 9 11 13 15 17 19
Sum = 100

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

Q5. Print a set of prime numbers from 1 to 30 using continue.

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

int main() {
    for (int i = 2; i <= 30; i++) {
        bool isPrime = true;

        for (int j = 2; j < i; j++) {
            if (i % j == 0) {
                isPrime = false;
                break;
            }
        }

        if (!isPrime)
            continue;

        cout << i << " ";
    }

    return 0;
}
```

Output

```
2 3 5 7 11 13 17 19 23 29

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

Q6. Write a program to perform function overloading.

main.cpp

```cpp
#include <iostream>
using namespace std;
class A{
    public:
    double dim(int l,int b){
    return l*b;
}
    double dim(int r){
    return 3.14*r*r;
}};
    int main(){
        A obj1;
        int l,b,r;
    cout<<"Enter the length and breadth:"<<endl;
        cin>>l>>b;
        cout<<"area:"<< obj1.dim(l,b)<<endl;
     cout<<"Enter the radius"<<endl;
       cin>>r;
cout<<"area:"<<obj1.dim(r)<<endl;
    return 0;
}
```

Output

```
Enter the length and breadth:
5 7
area:35
Enter the radius
6
area:113.04


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

Q7. Design a single C++ program to illustrate the following concepts of inheritance:

- Public derivation
- Private derivation
- Protected derivation

main.cpp

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

class Base {
protected:
    string name;
};

class PublicDerived : public Base {
public:
    void set(string n) { name = n; }
    void show() { cout << "Public: " << name << endl; }
};

class PrivateDerived : private Base {
public:
    void set(string n) { name = n; }
    void show() { cout << "Private: " << name << endl; }
};

class ProtectedDerived : protected Base {
public:
    void set(string n) { name = n; }
    void show() { cout << "Protected: " << name << endl; }
};

int main() {
    PublicDerived p; p.set("Alice"); p.show();
    PrivateDerived q; q.set("Bob"); q.show();
    ProtectedDerived r; r.set("Charlie"); r.show();
    return 0;
}
```

Output

```
Public: Alice
Private: Bob
Protected: Charlie


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

Q8 Write a program in c++ to use scope resolution operator.

```cpp
#include <iostream>
using namespace std;
int num =20;
int main(){
    int num=30;
    cout<<"local num:"<<num<<endl;
    cout<<"global num:"<<::num<<endl;
return 0;
}

```

Output

```
local num:30
global num:20


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

9. Write a program to print right angled triangle pyramid with '*'.

main.cpp

```cpp
#include <iostream>
using namespace std;
int main(){
    int i,j,n;
    cout<<"Enter number of rows:";
    cin>>n;
    for(i=1;i<=n;i++){
        for(j=1;j<=i;j++){
            cout<<"*";
        }
        cout<<endl;
    }
    return 0;
}

```

Output

```
Enter number of rows:8
*
**
***
****
*****
******
*******
********
```

Q10.Write a program in C++ to print an inverted right-angled triangle pyramid with '*'

```cpp
#include <iostream>
using namespace std;
int main() {
    int i, j, n;
    cout << "Enter number of rows: ";
    cin >> n;
    for (i = n; i >= 1; i--) {
        for (j = 1; j <= i; j++) {
            cout << "*";
        }
        cout << endl;
    }
 return 0;
}
```

Output

```
Enter number of rows: 5
*****
****
***
**
*
```

Q11. Print a right-angled triangle pyramid using numbers.

main.cpp

```
#include <iostream>
using namespace std;
int main() {
    int n;
    cout << "Enter number of rows: ";
    cin >> n;
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i; j++) {
            cout << j;
        }
        cout << endl;
    }
    return 0;
}
```

Output

```
Enter number of rows: 10
1
12
123
1234
12345
123456
1234567
12345678
123456789
12345678910
```

Q12. Design a basic calculator using switch.

main.cpp

```cpp
#include <iostream>
using namespace std;
int main() {
    char op;
    float a, b;
    cout << "Enter operator (+, -, *, /): ";
    cin >> op;
    cout << "Enter two numbers: ";
    cin >> a >> b;
    switch(op) {
        case '+': cout << "Result: " << a + b; break;
        case '-': cout << "Result: " << a - b; break;
        case '*': cout << "Result: " << a * b; break;
        case '/': cout << "Result: " << a / b; break;
        default: cout << "Invalid operator!";
    }
    return 0;
}
```

Output

```
Enter operator (+, -, *, /): /
Enter two numbers: 15 6
Result: 2.5

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

Q13. Check even or odd using goto.

main.cpp

```cpp
#include <iostream>
using namespace std;
int main() {
    int n;
    cout << "Enter a number: ";
    cin >> n;
    if (n % 2 == 0)
        goto even;
    else
        goto odd;
even:
    cout << "Even number";
    return 0;
odd:
    cout << "Odd number";
    return 0;
}
```

Output

```
Enter a number: 5
Odd number

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

Q14. Write a program to illustrate the use of:

- inline function
- function with default arguments

15. Write a program to read an employee's info from the user and print the same. Employee info includes employee ID using int, employee name using string, employee salary using class and object

main.cpp

Share Run

```cpp
#include <iostream>
using namespace std;
class Employee {
public:
    int id;
    string name;
    float salary;
    void input() {
        cout << "Enter Employee ID: ";
        cin >> id;
        cout << "Enter Employee Name: ";
        cin >> name;
        cout << "Enter Employee Salary: ";
        cin >> salary;
    }
    void display() {
        cout << "\nEmployee Details:\n";
        cout << "ID: " << id << endl;
        cout << "Name: " << name << endl;
        cout << "Salary: " << salary << endl;
    }
};
int main() {
    Employee emp;
    emp.input();
    emp.display();
    return 0;
}
```

Output

```
Enter Employee ID: 234
Enter Employee Name: rey
Enter Employee Salary: 23000

Employee Details:
ID: 234
Name: rey
Salary: 23000


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

Q16. Write a program to illustrate the use of friend function.

```cpp
// Online C++ compiler to run C++ program
    online
#include <iostream>
using namespace std;
class two;

class one
{
    private:
        int data1;
    public:
        one()
        {
            data1=100;
        }
        friend int accessboth(one,two);
};
class two
{
    private:
        int data2 ;
    public:
        two()
        {
            data2=20;
        }
        friend int accessboth(one,two);
};
int accessboth (one a, two b)
{
    return (a.data1 + b.data2);
}
int main() {
    // Write C++ code here
    one a;
    two b;
    cout<<accessboth(a,b)<<endl;
    return 0;
}
```

Output Clear

```
120


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

Q17. Use friend function to compare the speed of car object and truck object. Declare the function as friend of both classes.

```cpp
#include <iostream>
using namespace std;
class truck;
class car
{private:
        int speedcar;
    public:
        car()
        {speedcar = 25;}
    friend int compare(car , truck); };
class truck
{private:
        int speedtruck;
    public:
        truck(){ speedtruck =16; }
    friend int compare (car, truck); };
int compare (car c, truck t)
{if (c.speedcar > t.speedtruck)
        {cout<<"car speed greater"<<endl;}
    else
        {cout<<"truck speed greater"<<endl;}
    return 0;}
int main(){
    car c;
    truck t;
    cout<< compare (c,t)<< endl;
    return 0; }
```

```
car speed greater
0


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

Q18. Write a program to swap two numbers using-

a) Call by value b) Call by reference c) Call by reference without pointer.

```cpp
#include <iostream>
using namespace std;
void fun(int num) {
    num = num + 100;
    cout<<num<<endl;
}
int main() {
    int a = 65;
    fun(a);
    cout << "Value of a after fun: " << a
         << endl;
    return 0;
}

```

```
165
Value of a after fun: 65


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

```cpp
#include <iostream>
using namespace std;
void fun(int &num) {
    num = num + 100;
    cout<<num<<endl;
}
int main() {
    int a = 65;
    fun(a);
    cout << "Value of a after fun: " << a
         << endl;
    return 0;
}

```

```
165
Value of a after fun: 165


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

```cpp
#include <iostream>
using namespace std;
void swap(int &x, int &y) {
    int temp = x;
    x = y;
    y = temp;
}
int main() {
    int a = 43, b = 92;
    cout << "Before swapping: a = " << a
         << ", b = " << b << endl;
    swap(a, b);
    cout << "After swapping: a = " << a <<
         ", b = " << b << endl;
    return 0;
}
```

```
Before swapping: a = 43, b = 92
After swapping: a = 92, b = 43


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

Q19. Write a program to demonstrate operator overloading using the '+' operator to add two complex numbers.

```cpp
#include <iostream>
using namespace std;
class Complex {
    float real;
    float imag;
public:
    Complex(float r = 0, float i = 0) {
        real = r;
        imag = i;
    }
    Complex operator + (const Complex& c) {
        Complex temp;
        temp.real = real + c.real;
        temp.imag = imag + c.imag;
        return temp;
    }
    void display() {
        cout << real << " + " << imag << "i" << endl;
    }
};
int main() {
    Complex c1(3.5, 2.5);
    Complex c2(1.5, 4.5);
    Complex c3 = c1 + c2;
    cout << "Sum of complex numbers: ";
    c3.display();
    return 0;
}
```

```
Sum of complex numbers: 5 + 7i


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

Q20. Write a program to demonstrate both static and non-static members within a class.

```cpp
#include <iostream>
using namespace std;
class Student {
public:
   static int count;
   int count1;
   Student() {
      count++;
       count1--; }};
int Student::count = 0;
int main() {
   Student s1, s2, s3;
   cout << "Total students: " << Student
       ::count << endl;
   cout<<"non static: "<<s3.count1<<endl;
   return 0;}
```

```
Total students: 3
non static: -1


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

Q21. Write a program to demonstrate the concept of virtual function.

```cpp
#include <iostream>
using namespace std;
class Base {
public:
    virtual void show() {
        cout << "Base class function called" << endl;
    }};
class Derived : public Base {
public:
    void show() override {
        cout << "Derived class function called" << endl;
    }};
int main() {
    Base* ptr;
    Derived d;
    ptr = &d;
    ptr->show();
    return 0;
}
```

```
Derived class function called


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

Q22. Write a program to illustrate the use of try, catch and throw blocks (exception handling) in c++.

```cpp
#include <iostream>
using namespace std;
int main() {
    int a, b;
    cout << "Enter two numbers (dividend and divisor): ";
    cin >> a >> b;
    try {
        if (b == 0) {
            throw "Division by zero error!";
        }
        cout << "Result: " << a / b << endl;
    } catch (const char* msg) {
        cout << "Exception caught: " << msg << endl;
    }
    return 0;
}
```

```
Enter two numbers (dividend and divisor): 4 0
Exception caught: Division by zero error!


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

Q23. Write a basic program in java to implement class and object.

```java
class Car {
    String color = "Red";
    void drive() {
        System.out.println("The car is driving");
    }
    public static void main(String[] args) {
        Car myCar = new Car();
        System.out.println(myCar.color);
        myCar.drive();
    }
}
```

```
Red
The car is driving

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

Q24. Write a program in java to demonstrate the use of 'instanceof' keyword.

```java
class Animal {}
class Dog extends Animal {}

public class Main {
    public static void main(String[] args) {
        Animal a = new Dog();
        System.out.println(a instanceof Dog);
        System.out.println(a instanceof Animal);
        System.out.println(a instanceof Object);
    }
}
```

```
true
true
true

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

Q25. Write a program in java to demonstrate the 3 uses of super

keyword:

-to access parent class instance variables.
-to invoke parent class method.
-to invoke parent class constructor.

```java
class Parent {
    int value = 50;
    Parent() {
        System.out.println("Parent constructor called");
    }
    void show() {
        System.out.println("Parent method called");
    }
}
class Child extends Parent {
    int value = 100;
    Child() {
        super();
    }
    void display() {
        System.out.println("Parent value: " + super.value);
        super.show();
    }
}
public class Main {
    public static void main(String[] args) {
        Child c = new Child();
        c.display();
    }
}
```

```
Parent constructor called
Parent value: 50
Parent method called

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

Q26. Write a java program to illustrate the use of abstract class.

```java
abstract class Animal {
    abstract void sound();
    void eat() {
        System.out.println("This animal eats food");
    }
}
class Dog extends Animal {
    void sound() {
        System.out.println("Dog barks");
    }
}
public class Main {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.sound();
        d.eat();
    }
```

```
Dog barks
This animal eats food

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

Q27. Write a java program to illustrate the use of 'interface'.

```java
interface Animal {
    void sound();
}
class Cat implements Animal {
    public void sound() {
        System.out.println("Cat meows");
    }
}
public class Main {
    public static void main(String[] args) {
        Cat c = new Cat();
        c.sound();
    }
}
```

```
Cat meows

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

Q28. Write a Java program that demonstrates the use of the final keyword.

```java
final class Vehicle {
    final int speed = 100;
    final void display() {
        System.out.println("Speed is " + speed);
    }
}
public class Main {
    public static void main(String[] args) {
        Vehicle v = new Vehicle();
        v.display();
    }
}
```

```
Speed is 100

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

Q29. Write a Java program to implement Multithreading Using the Runnable Interface.

```java
class MyThread implements Runnable {
    public void run() {
        System.out.println("Thread is running");
    }
    public static void main(String[] args) {
        MyThread m = new MyThread();
        Thread t = new Thread(m);
        t.start();
    }
}
```

```
Thread is running

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

Q30. Write a Java program that demonstrates the use of exception handling mechanisms such as try, catch, finally, and throw to handle runtime errors effectively.

```java
public class Main {
    public static void main(String[] args) {
        try {
            int a = 10 / 0;
        } catch (ArithmeticException e) {
            System.out.println("Cannot divide by zero");
        } finally {
            System.out.println("Finally block executed");
        }
        try {
            throw new Exception("Custom exception");
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

```
Cannot divide by zero
Finally block executed
Custom exception

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