



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

DOMAIN WINTER CAMP WORKSHEET DAY-1

Student Name: Abhishek Thakur

Branch: BE-CSE

Semester: 5th

UID: 22BCS10686

Section/Group: FL_IOT_603 - A

Date: 19-12-2024

Problem-1:- (Very Easy)

1. Sum of Natural Numbers up to N.

Calculate the sum of all natural numbers from 1 to n, where n is a positive integer. Use the formula:

$$\text{Sum} = n \times (n+1) / 2$$

Take n as input and output the sum of natural numbers from 1 to n.

Source Code

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

int sumOfNaturalNumbers(int n) {
    return n * (n + 1) / 2;
}

int main() {
    int n = 10;
    int sum = sumOfNaturalNumbers(n);
    cout << "The sum of the first " << n << " natural numbers is: " << sum << endl;
    return 0;
}
```

Output

```
The sum of the first 10 natural numbers is: 55
```

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

Problem-2:- (Easy)

2. Count Digits in a Number

Count the total number of digits in a given number n. The number can be a positive integer. For example, for the number 12345, the count of digits is 5. For a number like 900000, the count of digits is 6. Given an integer n, your task is determining how many digits are present in n. This task will help you practice working with loops, number manipulation, and conditional logic.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Source Code

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

int countDigits(int n) {
    int count = 0;
    while (n != 0) {
        n /= 10;
        count++;
    }
    return count;
}

int main() {
    int n = 1234565;
    int digitCount = countDigits(n);
    cout << "The number of digits in " << n << " is: " << digitCount << endl;
    return 0;
}
```

Output

```
The number of digits in 1234565 is: 7
```

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

Problem-3:- (Medium)

3. Function Overloading for Calculating Area.

Write a program to calculate the area of different shapes using function overloading. Implement overloaded functions to compute the location of a circle, a rectangle, and a triangle.

Source Code

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

const double PI = 3.14159;

double areaCircle(double radius) {
    return PI * radius * radius;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}  
  
double areaRectangle(double length, double breadth) {  
    return length * breadth;  
}  
  
double areaTriangle(double base, double height) {  
    return 0.5 * base * height;  
}  
  
int main() {  
    cout << "Circle area: " << areaCircle(7.0) << endl;  
    cout << "Rectangle area: " << areaRectangle(5.0, 3.0) << endl;  
    cout << "Triangle area: " << areaTriangle(4.0, 6.0) << endl;  
    return 0;  
}
```

Output

```
Circle area: 153.938  
Rectangle area: 15  
Triangle area: 12
```

Problem-4:- (Hard)

4. Implement Polymorphism for Banking Transactions

Design a C++ program to simulate a banking system using polymorphism. Create a base class Account with a virtual method calculateInterest(). Use the derived classes SavingsAccount and CurrentAccount to implement specific interest calculation logic:

- SavingsAccount: Interest = Balance × Rate × Time.
- CurrentAccount: No interest, but includes a maintenance fee deduction.

Source Code

```
#include <iostream>  
using namespace std;  
  
class Account {  
protected:  
    int balance;  
  
public:  
    Account(int balance) {  
        this->balance = balance;  
    }  
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
virtual void calculateInterest() = 0; // Pure virtual function
};

class SavingsAccount : public Account {
private:
    double rate;
    int time;

public:
    SavingsAccount(int balance, double rate, int time) : Account(balance), rate(rate), time(time) {}

    void calculateInterest() override {
        double interest = balance * (rate / 100) * time;
        cout << "Interest for Savings Account: " << interest << endl;
    }
};

class CurrentAccount : public Account {
private:
    int monthlyFee;

public:
    CurrentAccount(int balance, int monthlyFee) : Account(balance), monthlyFee(monthlyFee) {}

    void calculateInterest() override {
        int yearlyFee = monthlyFee * 12;
        int remainingBalance = balance - yearlyFee;
        cout << "Balance after yearly maintenance fee in Current Account: " << remainingBalance << endl;
    }
};

int main() {
    int accountType;
    int balance;

    cout << "Enter Account Type (1 for Savings, 2 for Current): ";
    cin >> accountType;

    cout << "Enter Account Balance: ";
    cin >> balance;

    if (accountType == 1) {
        double rate;
        int time;

        cout << "Enter Interest Rate (%): ";
        cin >> rate;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
cout << "Enter Time (years): ";
cin >> time;

SavingsAccount savingsAccount(balance, rate, time);
savingsAccount.calculateInterest();
}
else if (accountType == 2) {
    int monthlyFee;

    cout << "Enter Monthly Maintenance Fee: ";
    cin >> monthlyFee;

    CurrentAccount currentAccount(balance, monthlyFee);
    currentAccount.calculateInterest();
}
else {
    cout << "Invalid Account Type" << endl;
}

return 0;
}
```

Output

```
Enter Account Type (1 for Savings, 2 for Current): 1
Enter Account Balance: 20000
Enter Interest Rate (%): 5
Enter Time (years): 12
Interest for Savings Account: 12000
```

Problem-5:- (Very Hard)

5. Implementing Polymorphism for Shape Hierarchies.

Create a C++ program to simulate an employee management system using hierarchical inheritance. Design a base class Employee that stores basic details (name, ID, and salary). Create two derived classes: Manager: Add and calculate bonuses based on performance ratings. Developer: Add and calculate overtime compensation based on extra hours worked.

The program should allow input for both types of employees and display their total earnings.

Source Code:

```
#include <iostream>
```

```
#include <string>
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
using namespace std;
```

```
class Employee {
```

```
public:
```

```
    string name;
```

```
    int ID;
```

```
    int salary;
```

```
    Employee(string n, int i, int s) : name(n), ID(i), salary(s) {
```

```
        if (salary < 10000 || salary > 1000000) {
```

```
            cout << "Invalid salary. Salary must be between 10000 and 1000000." << endl;
```

```
            exit(1);
```

```
        }
```

```
    }
```

```
    virtual int getTotalEarnings() {
```

```
        return salary;
```

```
    }
```

```
};
```

```
class Manager : public Employee {
```

```
public:
```

```
    int performanceRating;
```

```
    Manager(string n, int i, int s, int pr) : Employee(n, i, s), performanceRating(pr) {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
if (performanceRating < 1 || performanceRating > 5) {  
    cout << "Invalid performance rating. Rating must be between 1 and 5." << endl;  
    exit(1);  
}  
}
```

```
int getTotalEarnings() override {  
    int bonus = 0;  
    switch (performanceRating) {  
        case 1:  
            bonus = 0;  
            break;  
        case 2:  
            bonus = salary * 0.05;  
            break;  
        case 3:  
            bonus = salary * 0.10;  
            break;  
        case 4:  
            bonus = salary * 0.15;  
            break;  
        case 5:  
            bonus = salary * 0.20;  
            break;  
    }  
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

return salary + bonus;

}

};

class Developer : public Employee {

public:

int extraHours;

Developer(string n, int i, int s, int eh) : Employee(n, i, s), extraHours(eh) {}

int getTotalEarnings() override {

int overtimePay = extraHours * (salary / 200); // Assuming 200 working hours per month

return salary + overtimePay;

}

};

int main() {

int employeeType, ID, salary, performanceRating, extraHours;

string name;

cout << "Enter Employee Type (1 for Manager, 2 for Developer): ";

cin >> employeeType;

cout << "Enter Name: ";

cin >> name;



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
cout << "Enter ID: ";
```

```
cin >> ID;
```

```
cout << "Enter Salary: ";
```

```
cin >> salary;
```

```
if (employeeType == 1) {
```

```
    cout << "Enter Performance Rating (1-5): ";
```

```
    cin >> performanceRating;
```

```
    Manager manager(name, ID, salary, performanceRating);
```

```
    cout << "Total Earnings for Manager " << manager.name << ": " <<
manager.getTotalEarnings() << endl;
```

```
} else if (employeeType == 2) {
```

```
    cout << "Enter Extra Hours Worked: ";
```

```
    cin >> extraHours;
```

```
    Developer developer(name, ID, salary, extraHours);
```

```
    cout << "Total Earnings for Developer " << developer.name << ": " <<
developer.getTotalEarnings() << endl;
```

```
} else {
```

```
    cout << "Invalid Employee Type." << endl;
```

```
}
```

```
return 0;
```

```
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Output

```
Enter Employee Type (1 for Manager, 2 for Developer): 1
Enter Name: Abhishek
Enter ID: 2004
Enter Salary: 2000000
Enter Performance Rating (1-5): 5
```

Problem-6:- (Hard)

6. Implementing Polymorphism for Shape Hierarchies.

Write a program to demonstrate runtime polymorphism in C++ using a base class Shape and derived classes Circle, Rectangle, and Triangle. The program should use virtual functions to calculate and print the area of each shape based on user input.

Source Code:

```
#include <iostream>

using namespace std;

class Shape {
public:
    virtual void calculateArea() = 0; // Pure virtual function
};

class Circle : public Shape {
private:
    double radius;
public:
    Circle(double radius) {
        this->radius = radius;
    }
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
void calculateArea() override {  
  
    double area = 3.14159 * radius * radius;  
  
    cout << "Area of Circle: " << area << endl;  
  
}  
  
};  
  
class Rectangle : public Shape {  
  
private:  
  
    double length, breadth;  
  
public:  
  
    Rectangle(double length, double breadth) {  
  
        this->length = length;  
  
        this->breadth = breadth;  
  
    }  
  
  
    void calculateArea() override {  
  
        double area = length * breadth;  
  
        cout << "Area of Rectangle: " << area << endl;  
  
    }  
  
};  
  
class Triangle : public Shape {  
  
private:  
  
    double base, height;  
  
public:  
  
    Triangle(double base, double height) {  
  
        this->base = base;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

this->height = height;

}

void calculateArea() override {

double area = 0.5 * base * height;

cout << "Area of Triangle: " << area << endl;

}

};

int main() {

double radius, length, breadth, base, height;

cout << "Enter radius of the circle: ";

cin >> radius;

Shape* circle = new Circle(radius);

circle->calculateArea();

cout << "Enter length and breadth of the rectangle: ";

cin >> length >> breadth;

Shape* rectangle = new Rectangle(length, breadth);

rectangle->calculateArea();

cout << "Enter base and height of the triangle: ";

cin >> base >> height;

Shape* triangle = new Triangle(base, height);

triangle->calculateArea();

delete circle;

delete rectangle;

delete triangle;



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
return 0;
```

```
}
```

Output

```
Enter radius of the circle: 25
Area of Circle: 1963.49
Enter length and breadth of the rectangle: 3
5
Area of Rectangle: 15
Enter base and height of the triangle: 5
68
Area of Triangle: 170
```

Problem-7:- (Easy)

7. Given an integer n, print "Prime" if the number is prime, or "Not Prime" if it is not.

Source Code:

```
#include <iostream>
```

```
using namespace std;
```

```
bool isPrime(int n) {
```

```
    if (n <= 1) return false;
```

```
    if (n == 2 || n == 3) return true;
```

```
    if (n % 2 == 0 || n % 3 == 0) return false;
```

```
    return true;
```

```
}
```

```
int main() {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
int n;  
  
cout << "Enter a number: ";  
  
cin >> n;  
  
if (isPrime(n)) {  
    cout << "Prime" << endl;  
} else {  
    cout << "Not Prime" << endl;  
}  
  
return 0;  
}
```

Output

```
Enter a number: 5  
Prime
```

Problem-8:- (Easy)

8. Given an integer n, print all odd numbers from 1 to n, inclusive.

Source Code:

```
#include <iostream>  
  
using namespace std;  
  
int main() {  
    int n;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
cout << "Enter a number: ";
```

```
cin >> n;
```

```
for (int i = 1; i <= n; i += 2) {
```

```
    cout << i << " ";
```

```
}
```

```
cout << endl;
```

```
return 0;
```

```
}
```

Output

```
Enter a number: 13
1 3 5 7 9 11 13
```

Problem-9:- (Hard)

9. Print the sum of all odd numbers from 1 to n.

Source Code:

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {
```

```
    int n, sum = 0;
```

```
    cout << "Enter a number: ";
```

```
    cin >> n;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
for (int i = 1; i <= n; i += 2) {  
  
    sum += i;  
  
}  
  
cout << "Sum of odd numbers from 1 to " << n << " is: " << sum << endl;  
  
return 0;  
  
}
```

Output

```
Enter a number: 8  
Sum of odd numbers from 1 to 8 is: 16
```

Problem-10:- (Easy)

10. Given an integer n, print the multiplication table of n from $1 \times n$ to $10 \times n$.

Source Code:

```
#include <iostream>  
  
using namespace std;  
  
int main() {  
  
    int n;  
  
    cout << "Enter a number: ";  
  
    cin >> n;  
  
  
    for (int i = 1; i <= 10; i++) {
```




DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
cout << i << " × " << n << " = " << i * n << endl;
```

```
}
```

```
return 0;
```

```
}
```

Output

```
Enter a number: 5
1 × 5 = 5
2 × 5 = 10
3 × 5 = 15
4 × 5 = 20
5 × 5 = 25
6 × 5 = 30
7 × 5 = 35
8 × 5 = 40
9 × 5 = 45
10 × 5 = 50
```

Problem-11:- (Easy)

11. Given an integer n, print the number with its digits in reverse order.

Source Code:

```
#include <iostream>
```

```
using namespace std;
```

```
int main() {
```

```
    int n, reversed = 0;
```

```
    cout << "Enter a number: ";
```

```
    cin >> n;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
while (n != 0) {  
  
    int digit = n % 10;  
  
    reversed = reversed * 10 + digit;  
  
    n /= 10;  
  
}  
  
cout << "Reversed number: " << reversed << endl;  
  
return 0;  
  
}
```

Output

```
Enter a number: 1234  
Reversed number: 4321
```

Problem-12:- (Easy)

12. Given an integer n, find and print the largest digit in n.

Source Code:

```
#include <iostream>  
  
using namespace std;  
  
int main() {  
  
    int n, largestDigit = 0;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
cout << "Enter a number: ";
```

```
cin >> n;
```

```
while (n != 0) {
```

```
    int digit = n % 10;
```

```
    if (digit > largestDigit) {
```

```
        largestDigit = digit;
```

```
    }
```

```
    n /= 10;
```

```
}
```

```
cout << "The largest digit is: " << largestDigit << endl;
```

```
return 0;
```

```
}
```

Output

```
Enter a number: 678
The largest digit is: 8
```

Problem-13:- (Easy)

13. Given an integer n, print "Palindrome" if the number is a palindrome, otherwise print "Not Palindrome".



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Source Code:

```
#include <iostream>

using namespace std;

int main() {

    int n, original, reversed = 0;

    cout << "Enter a number: ";

    cin >> n;

    original = n; // Store the original number

    while (n != 0) {

        int digit = n % 10;

        reversed = reversed * 10 + digit;

        n /= 10;

    }

    if (original == reversed) {

        cout << "Palindrome" << endl;

    } else {

        cout << "Not Palindrome" << endl;

    }

    return 0;

}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Output

```
Enter a number: 234  
Not Palindrome
```

Problem-14:- (Easy)

14. Given an integer n, find and print the sum of its digits.

Source Code:

```
#include <iostream>

using namespace std;

int main() {

    int n, sum = 0;

    cout << "Enter a number: ";

    cin >> n;

    while (n != 0) {

        int digit = n % 10;

        sum += digit;

        n /= 10;

    }

    cout << "Sum of digits: " << sum << endl;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

return 0;

}

Output

```
Enter a number: 3245
Sum of digits: 14
```

Problem-15:- (Medium)

15. Write a program that demonstrates encapsulation by creating a class Employee. The class should have private attributes to store:

Employee ID.

Employee Name.

Employee Salary.

Provide public methods to set and get these attributes, and a method to display all details of the employee.

Source Code:

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
class Employee {
```

```
private:
```

```
    int employeeID;
```

```
    string employeeName;
```

```
    double employeeSalary;
```

```
public:
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

// Setter methods

```
void setEmployeeID(int id) {  
    employeeID = id;  
}
```

```
void setEmployeeName(string name) {  
    employeeName = name;  
}
```

```
void setEmployeeSalary(double salary) {  
    employeeSalary = salary;  
}
```

// Getter methods

```
int getEmployeeID() {  
    return employeeID;  
}
```

```
string getEmployeeName() {  
    return employeeName;  
}
```

```
double getEmployeeSalary() {  
    return employeeSalary;  
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
// Method to display employee details

void displayEmployeeDetails() {

    cout << "Employee ID: " << employeeID << endl;

    cout << "Employee Name: " << employeeName << endl;

    cout << "Employee Salary: " << employeeSalary << endl;

}

};


int main() {

    Employee emp;


    // Setting employee details

    emp.setEmployeeID(101);

    emp.setEmployeeName("John Doe");

    emp.setEmployeeSalary(50000.75);


    // Displaying employee details

    emp.displayEmployeeDetails();


    return 0;

}
```




DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Output

```
Employee ID: 101  
Employee Name: John  
Employee Salary: 50000.8
```

Problem-16:- (Medium)

16. Write a program to demonstrate runtime polymorphism in C++ using a base class Shape and derived classes Circle, Rectangle, and Triangle. The program should use virtual functions to calculate and print the area of each shape based on user input.

Source Code:

```
#include <iostream>

using namespace std;

class Shape {
public:
    virtual void calculateArea() = 0; // Pure virtual function
};

class Circle : public Shape {
private:
    double radius;
public:
    Circle(double r) : radius(r) {}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
void calculateArea() override {  
  
    double area = 3.14159 * radius * radius;  
  
    cout << "Area of Circle: " << area << endl;  
  
}  
  
};
```

```
class Rectangle : public Shape {  
  
private:  
  
    double length, breadth;  
  
public:  
  
    Rectangle(double l, double b) : length(l), breadth(b) {}  
  
    void calculateArea() override {  
  
        double area = length * breadth;  
  
        cout << "Area of Rectangle: " << area << endl;  
  
    }  
  
};
```

```
class Triangle : public Shape {  
  
private:  
  
    double base, height;  
  
public:  
  
    Triangle(double b, double h) : base(b), height(h) {}  
  
    void calculateArea() override {
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
double area = 0.5 * base * height;
```

```
cout << "Area of Triangle: " << area << endl;
```

```
}
```

```
};
```

```
int main() {
```

```
    double radius, length, breadth, base, height;
```

```
    // Input values for shapes
```

```
    cout << "Enter radius of the circle: ";
```

```
    cin >> radius;
```

```
    cout << "Enter length and breadth of the rectangle: ";
```

```
    cin >> length >> breadth;
```

```
    cout << "Enter base and height of the triangle: ";
```

```
    cin >> base >> height;
```

```
    // Creating objects for each shape
```

```
    Circle circle(radius);
```

```
    Rectangle rectangle(length, breadth);
```

```
    Triangle triangle(base, height);
```

```
    // Using polymorphism to calculate area
```

```
    Shape* shapePtr;
```

```
    shapePtr = &circle;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.
shapePtr->calculateArea();

shapePtr = &rectangle;

shapePtr->calculateArea();

shapePtr = ▵

shapePtr->calculateArea();

return 0;

}

Output

```
Enter radius of the circle: 23
Enter length and breadth of the rectangle: 12
2
Enter base and height of the triangle: 2
4
Area of Circle: 1661.9
Area of Rectangle: 24
Area of Triangle: 4
```

Problem-17:- (Hard)

17. Implement matrix operations in C++ using function overloading. Write a function operate() that can perform:
Matrix Addition for matrices of the same dimensions.
Matrix Multiplication where the number of columns of the first matrix equals the number of rows of the second matrix.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Source Code:

```
#include <iostream>

using namespace std;

class Matrix {

private:

    int rows, cols;

    int** mat;

public:

    // Constructor to create a matrix of given size
    Matrix(int r, int c) : rows(r), cols(c) {

        mat = new int*[rows];

        for (int i = 0; i < rows; ++i) {

            mat[i] = new int[cols]();

        }

    }

    // Destructor to free dynamically allocated memory
    ~Matrix() {

        for (int i = 0; i < rows; ++i) {

            delete[] mat[i];

        }

        delete[] mat;

    }

}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

// Function to input matrix elements

```
void input() {  
  
    cout << "Enter elements of the matrix (" << rows << "x" << cols << "):" << endl;  
  
    for (int i = 0; i < rows; ++i) {  
  
        for (int j = 0; j < cols; ++j) {  
  
            cin >> mat[i][j];  
  
        }  
  
    }  
  
}
```

// Function to display matrix

```
void display() {  
  
    for (int i = 0; i < rows; ++i) {  
  
        for (int j = 0; j < cols; ++j) {  
  
            cout << mat[i][j] << " ";  
  
        }  
  
        cout << endl;  
  
    }  
  
}
```

// Function to perform matrix addition (overloaded)

```
void operate(const Matrix& m) {  
  
    if (this->rows != m.rows || this->cols != m.cols) {  
  
        cout << "Matrix addition is not possible. Matrices must have the same dimensions." << endl;  
  
    }  
  
}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
        return;

    }

    cout << "Matrix Addition Result:" << endl;

    for (int i = 0; i < rows; ++i) {

        for (int j = 0; j < cols; ++j) {

            mat[i][j] += m.mat[i][j];

        }

    }

    display();

}

// Function to perform matrix multiplication (overloaded)

void operate(const Matrix& m, bool multiply) {

    if (this->cols != m.rows) {

        cout << "Matrix multiplication is not possible. The number of columns of the first matrix
must equal the number of rows of the second matrix." << endl;

        return;

    }

    Matrix result(this->rows, m.cols); // Resultant matrix of size rows x cols

    for (int i = 0; i < this->rows; ++i) {

        for (int j = 0; j < m.cols; ++j) {

            result.mat[i][j] = 0;

            for (int k = 0; k < this->cols; ++k) {

                result.mat[i][j] += this->mat[i][k] * m.mat[k][j];

            }

        }

    }

}
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
}

cout << "Matrix Multiplication Result:" << endl;

result.display();

}

};

int main() {

    int r1, c1, r2, c2;

    // Input for matrix 1

    cout << "Enter rows and columns for Matrix 1: ";

    cin >> r1 >> c1;

    Matrix matrix1(r1, c1);

    matrix1.input();

    // Input for matrix 2

    cout << "Enter rows and columns for Matrix 2: ";

    cin >> r2 >> c2;

    Matrix matrix2(r2, c2);

    matrix2.input();

    // Matrix Addition

    matrix1.operate(matrix2); // Calling operate() for addition

    // Matrix Multiplication
```




DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
matrix1.operate(matrix2, true); // Calling operate() for multiplication
```

```
return 0;
```

```
}
```

Output

```
Enter rows and columns for Matrix 1: 2
2
Enter elements of the matrix (2x2):
3
4
5
6
Enter rows and columns for Matrix 2: 2
2
Enter elements of the matrix (2x2):
5
6
8
9
Matrix Addition Result:
8 10
13 15
Matrix Multiplication Result:
120 138
185 213
```

Problem-18:- (Hard)

18. Create a C++ program using multiple inheritance to simulate a library system. Design two base classes:
Book to store book details (title, author, and ISBN).

Borrower to store borrower details (name, ID, and borrowed book).

Create a derived class Library that inherits from both Book and Borrower. Use this class to track the borrowing and returning of books.



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Source Code:

```
#include <iostream>

#include <string>

using namespace std;

class Book {

protected:

    string title;

    string author;

    int ISBN;

public:

    // Constructor for Book

    Book(string t, string a, int isbn) : title(t), author(a), ISBN(isbn) { }

    // Display book details

    void displayBook() {

        cout << "Title: " << title << endl;

        cout << "Author: " << author << endl;

        cout << "ISBN: " << ISBN << endl;

    }

};

class Borrower {

protected:
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

string name;

int ID;

public:

// Constructor for Borrower

Borrower(string n, int id) : name(n), ID(id) {}

// Display borrower details

void displayBorrower() {

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

 cout << "Borrower ID: " << ID << endl;

}

};

class Library : public Book, public Borrower {

private:

 bool isBookBorrowed;

public:

// Constructor for Library (inherits from both Book and Borrower)

Library(string t, string a, int isbn, string n, int id)

 : Book(t, a, isbn), Borrower(n, id), isBookBorrowed(false) {}

// Borrow book method

void borrowBook() {



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
if (isBookBorrowed) {
```

```
    cout << "Sorry, the book is already borrowed." << endl;
```

```
} else {
```

```
    isBookBorrowed = true;
```

```
    cout << "Borrower " << name << " (ID: " << ID << ") has borrowed \"" << title
```

```
        << "\" by " << author << " (ISBN: " << ISBN << ")." << endl;
```

```
}
```

```
}
```

```
// Return book method
```

```
void returnBook() {
```

```
    if (!isBookBorrowed) {
```

```
        cout << "The book was not borrowed." << endl;
```

```
    } else {
```

```
        isBookBorrowed = false;
```

```
        cout << "Borrower " << name << " (ID: " << ID << ") has returned \"" << title
```

```
            << "\" by " << author << " (ISBN: " << ISBN << ")." << endl;
```

```
    }
```

```
}
```

```
};
```

```
int main() {
```

```
    string title, author, borrowerName;
```

```
    int ISBN, borrowerID, action;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

// Input book details

```
cout << "Enter Book Details:" << endl;
```

```
cout << "Title: ";
```

```
getline(cin, title);
```

```
cout << "Author: ";
```

```
getline(cin, author);
```

```
cout << "ISBN: ";
```

```
cin >> ISBN;
```

// Input borrower details

```
cin.ignore(); // To ignore the newline character left by cin
```

```
cout << "\nEnter Borrower Details:" << endl;
```

```
cout << "Name: ";
```

```
getline(cin, borrowerName);
```

```
cout << "ID: ";
```

```
cin >> borrowerID;
```

// Create Library object

```
Library lib(title, author, ISBN, borrowerName, borrowerID);
```

// Input action type

```
cout << "\nEnter Action Type:" << endl;
```

```
cout << "1 to Borrow a Book" << endl;
```

```
cout << "2 to Return a Book" << endl;
```

```
cin >> action;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
// Perform action based on user input

if (action == 1) {

    lib.borrowBook();

} else if (action == 2) {

    lib.returnBook();

} else {

    cout << "Invalid action!" << endl;

}

return 0;

}
```

Output

```
Enter Book Details:
Title: C++
Author: DEV
ISBN: 2344

Enter Borrower Details:
Name: Abhi
ID: 2345

Enter Action Type:
1 to Borrow a Book
2 to Return a Book
1
Borrower Abhi (ID: 2345) has borrowed "C++" by DEV (ISBN: 2344).
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

Problem-19:- (Very Hard)

Create a C++ program that demonstrates **function overloading** to calculate the area of different geometric shapes. Implement three overloaded functions named calculateArea that compute the area for the following shapes:

Circle: Accepts the radius.

Rectangle: Accepts the length and breadth.

Triangle: Accepts the base and height.

Additionally, use a menu-driven program to let the user choose the type of shape and input the respective parameters. Perform necessary validations on the input values.

Source Code:

```
#include <iostream>
```

```
#include <cmath>
```

```
using namespace std;
```

```
// Function to calculate the area of a circle
```

```
double calculateArea(double radius) {
```

```
    if (radius <= 0) {
```

```
        cout << "Invalid input. Radius must be positive." << endl;
```

```
        return -1;
```

```
    }
```

```
    return 3.14159 * radius * radius;
```

```
}
```

```
// Function to calculate the area of a rectangle
```

```
double calculateArea(double length, double breadth) {
```

```
    if (length <= 0 || breadth <= 0) {
```

```
        cout << "Invalid input. Length and breadth must be positive." << endl;
```

```
        return -1;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

}

return length * breadth;

}

// Function to calculate the area of a triangle

double calculateArea(double base, double height) {

if (base <= 0 || height <= 0) {

cout << "Invalid input. Base and height must be positive." << endl;

return -1;

}

return 0.5 * base * height;

}

int main() {

int choice;

cout << "Choose a shape:" << endl;

cout << "1. Circle" << endl;

cout << "2. Rectangle" << endl;

cout << "3. Triangle" << endl;

cout << "Enter your choice (1-3): ";

cin >> choice;

if (choice == 1) {

double radius;

cout << "Enter the radius of the circle: ";



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
cin >> radius;

double area = calculateArea(radius);

if (area != -1) {

    cout << "Shape: Circle" << endl;

    cout << "Radius: " << radius << endl;

    cout << "Area: " << area << endl;

}

} else if (choice == 2) {

    double length, breadth;

    cout << "Enter the length and breadth of the rectangle: ";

    cin >> length >> breadth;

    double area = calculateArea(length, breadth);

    if (area != -1) {

        cout << "Shape: Rectangle" << endl;

        cout << "Length: " << length << endl;

        cout << "Breadth: " << breadth << endl;

        cout << "Area: " << area << endl;

    }

} else if (choice == 3) {

    double base, height;

    cout << "Enter the base and height of the triangle: ";

    cin >> base >> height;

    double area = calculateArea(base, height);

    if (area != -1) {

        cout << "Shape: Triangle" << endl;
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
cout << "Base: " << base << endl;
```

```
cout << "Height: " << height << endl;
```

```
cout << "Area: " << area << endl;
```

```
}
```

```
} else {
```

```
    cout << "Invalid choice. Please select a valid shape type." << endl;
```

```
}
```

```
return 0;
```

```
}
```

Output

```
Choose a shape:
1. Circle
2. Rectangle
3. Triangle
Enter your choice (1-3): 2
Enter the length and breadth of the rectangle: 8.0 4.5
Shape: Rectangle
Length: 8
Breadth: 4.5
Area: 36
|
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.