Name = Harsh Raj Choudhary

UID = 22BCS11231

Sec/Group = KPIT 901-B

Problem 1 :

main.cpp — Share — Run — Output

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

namespace MathOperations {
    void calculateSum() {
        int n;
        cout << "Enter a positive integer (n): ";
        cin >> n;

        if (n > 0) {
            int sum = n * (n + 1) / 2;
            cout << "The sum of natural numbers from 1 to " << n << " is: " <<
                sum << endl;
        } else {
            cout << "Please enter a positive integer." << endl;
        }
    }
}

int main() {
    MathOperations::pcalculateSum();
    return 0;
}
```

```
Enter a positive integer (n): 4
The sum of natural numbers from 1 to 4 is: 10


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

Problem 2 :

main.cpp — Share — Run — Output

```cpp
#include <bits/stdc++.h>
namespace Geometry {
    double area(double radius) {
        return M_PI * radius * radius;
    }
    double area(double length, double width) {
        return length * width;
    }

    double area(double base, double height, bool isTriangle) {
        return 0.5 * base * height;
    }
}
int main() {
    using namespace std;
    using namespace Geometry;
    double radius, length, width, base, height;
    cout << "Enter radius of the circle: ";
    cin >> radius;
    cout << "Area of the circle: " << area(radius) << endl;
    cout << "Enter length and width of the rectangle: ";
    cin >> length >> width;
    cout << "Area of the rectangle: " << area(length, width) << endl;
    cout << "Enter base and height of the triangle: ";
    cin >> base >> height;
    cout << "Area of the triangle: " << area(base, height, true) << endl;
    return 0;
}
```

```
Enter radius of the circle: 4
Area of the circle: 50.2655
Enter length and width of the rectangle: 3
4
Area of the rectangle: 12
Enter base and height of the triangle: 2
4
Area of the triangle: 4


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

Problem 3 :

main.cpp Share Run

```cpp
#include <iostream>
using namespace std;
class Matrix {
public:
    int rows, cols;
    int **data;
    Matrix(int r, int c) : rows(r), cols(c) {
        data = new int*[rows];
        for(int i = 0; i < rows; ++i) {
            data[i] = new int[cols];
        }
    }
    ~Matrix() {
        for(int i = 0; i < rows; ++i) {
            delete[] data[i];
        }
        delete[] data;
    }
    void input() {
        cout << "Enter elements of the matrix (" << rows << " x " << cols << "
            ):\n";
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                cin >> data[i][j];
            }
        }
    }
```

main.cpp Share Run

```cpp
    void display() const {
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                cout << data[i][j] << " ";
            }
            cout << endl;
        }
    }
    Matrix add(const Matrix& mat) {
        if (rows != mat.rows || cols != mat.cols) {
            throw invalid_argument("Matrix dimensions must match for addition
                .");
        }
        Matrix result(rows, cols);
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < cols; ++j) {
                result.data[i][j] = data[i][j] + mat.data[i][j];
            }
        }
        return result;
    }
    Matrix multiply(const Matrix& mat) {
        if (cols != mat.rows) {
            throw invalid_argument("Number of columns of the first matrix must
                equal number of rows of the second matrix for multiplication."
                );
        }
```

main.cpp

Share Run

```
        }
        Matrix result(rows, mat.cols);
        for (int i = 0; i < rows; ++i) {
            for (int j = 0; j < mat.cols; ++j) {
                result.data[i][j] = 0;
                for (int k = 0; k < cols; ++k) {
                    result.data[i][j] += data[i][k] * mat.data[k][j];
                }
            }
        }
        return result;
    }
};
int main() {
    int rows1, cols1, rows2, cols2;
    cout << "Enter rows and columns of first matrix: ";
    cin >> rows1 >> cols1;
    Matrix mat1(rows1, cols1);
    mat1.input();
    cout << "Enter rows and columns of second matrix: ";
    cin >> rows2 >> cols2;
    Matrix mat2(rows2, cols2);
    mat2.input();
    try {
        if (rows1 == rows2 && cols1 == cols2) {
            Matrix sum = mat1.add(mat2);
            cout << "Matrix Addition Result:\n";
            sum.display();
        } else {
            cout << "Matrix dimensions do not match for addition." << endl;
        }
        if (cols1 == rows2) {
            Matrix product = mat1.multiply(mat2);
            cout << "Matrix Multiplication Result:\n";
            product.display();
        } else {
            cout << "Matrix dimensions do not match for multiplication." << endl;
        }
    } catch (const exception& e) {
        cout << "Error: " << e.what() << endl;
    }
    return 0;
}
```

Output

```
Enter rows and columns of first matrix: 2 2
Enter elements of the matrix (2 x 2):
2 3
4 5
Enter rows and columns of second matrix: 2 2
Enter elements of the matrix (2 x 2):
5 6
7 8
Matrix Addition Result:
7 9
11 13
Matrix Multiplication Result:
31 36
55 64
```

Problem 4 :

main.cpp Share Run

```cpp
#include <bits/stdc++.h>
using namespace std;
class Shape {
public:
    virtual double getArea() = 0;
};
class Rectangle : public Shape {
private:
    double length;
    double breadth;
public:
    Rectangle(double l, double b) : length(l), breadth(b) {}
    double getArea() override {
        return length * breadth;
    }
};
class Circle : public Shape {
private:
    double radius;
public:
    Circle(double r) : radius(r) {}
    double getArea() override {
        return M_PI * radius * radius;
    }
};
class Triangle : public Shape {
private:
    double base;
    double height;
public:
    Triangle(double b, double h) : base(b), height(h) {}
    double getArea() override {
        return 0.5 * base * height;
    }
};
int main() {
    Shape* shape1 = new Rectangle(5.0, 3.0);
    Shape* shape2 = new Circle(4.0);
    Shape* shape3 = new Triangle(6.0, 4.0);
    cout << "Area of Rectangle: " << shape1->getArea() << endl;
    cout << "Area of Circle: " << shape2->getArea() << endl;
    cout << "Area of Triangle: " << shape3->getArea() << endl;
    delete shape1;
    delete shape2;
    delete shape3;
    return 0;
}
```

```
Output

Area of Rectangle: 15
Area of Circle: 50.2655
Area of Triangle: 12
```

Problem 5 :

main.cpp

Share Run

```cpp
#include <bits/stdc++.h>
using namespace std;
class Book {
protected:
    string title;
    string author;
    string ISBN;
    Book(string t, string a, string isbn) : title(t), author(a), ISBN(isbn) {}
    void displayBookInfo() {
        cout << "Book Title: " << title << endl;
        cout << "Author: " << author << endl;
        cout << "ISBN: " << ISBN << endl;
    }
};
class Borrower {
protected:
    string name;
    int id;
    string borrowedBook;
public:
    Borrower(string n, int i) : name(n), id(i), borrowedBook("") {}
    void displayBorrowerInfo() {
        cout << "Borrower Name: " << name << endl;
        cout << "Borrower ID: " << id << endl;
        if (!borrowedBook.empty()) {
            cout << "Borrowed Book: " << borrowedBook << endl;
        } else {
            cout << "No book borrowed." << endl;
        }
    }
```

main.cpp

Share Run

```
    void borrowBook(string bookTitle) {
        borrowedBook = bookTitle;
        cout << name << " has borrowed the book: " << bookTitle << endl;
    }
    void returnBook() {
        if (!borrowedBook.empty()) {
            cout << name << " has returned the book: " << borrowedBook << endl
                ;
            borrowedBook = "";
        } else {
            cout << name << " has no borrowed books to return." << endl;
        }
    }
};
class Library : public Book, public Borrower {
public:
    Library(string bookTitle, string bookAuthor, string bookISBN, string
        borrowerName, int borrowerId)
        : Book(bookTitle, bookAuthor, bookISBN), Borrower(borrowerName,
            borrowerId) {}
    void displayLibraryInfo() {
        displayBookInfo();
        displayBorrowerInfo();
    }
};
```

```
int main() {
    Library library1("The Great Gatsby", "F. Scott Fitzgerald",
        "9780743273565", "John Doe", 101);
    library1.displayLibraryInfo();
    library1.borrowBook("The Great Gatsby");
    library1.displayLibraryInfo();
    library1.returnBook();
    library1.displayLibraryInfo();
    return 0;
}
```

Output

```
Book Title: The Great Gatsby
Author: F. Scott Fitzgerald
ISBN: 9780743273565
Borrower Name: John Doe
Borrower ID: 101
No book borrowed.
John Doe has borrowed the book: The Great Gatsby
Book Title: The Great Gatsby
Author: F. Scott Fitzgerald
ISBN: 9780743273565
Borrower Name: John Doe
Borrower ID: 101
Borrowed Book: The Great Gatsby
John Doe has returned the book: The Great Gatsby
Book Title: The Great Gatsby
Author: F. Scott Fitzgerald
ISBN: 9780743273565
Borrower Name: John Doe
Borrower ID: 101
No book borrowed.
```