

Name: Diksha Sharma

UID: 22BCS12218

Sec/Group = KPIT 901-B

Problem 1 :

main.cpp	Output
<pre>1 #include <iostream> 2 using namespace std; 3 4 namespace MathOperations { 5 void calculateSum() { 6 int n; 7 cout << "Enter a positive integer (n): "; 8 cin >> n; 9 10 if (n > 0) { 11 int sum = n * (n + 1) / 2; 12 cout << "The sum of natural numbers from 1 to " << n << " is: " << sum << endl; 13 } else { 14 cout << "Please enter a positive integer." << endl; 15 } 16 } 17 } 18 19 int main() { 20 MathOperations::calculateSum(); 21 return 0; 22 } 23</pre>	<pre>Enter a positive integer (n): 4 The sum of natural numbers from 1 to 4 is: 10 === Code Execution Successful ===</pre>

Problem 2 :

main.cpp	Output
<pre>1 #include <bits/stdc++.h> 2 namespace Geometry { 3 double area(double radius) { 4 return M_PI * radius * radius; 5 } 6 double area(double length, double width) { 7 return length * width; 8 } 9 10 double area(double base, double height, bool isTriangle) { 11 return 0.5 * base * height; 12 } 13 } 14 int main() { 15 using namespace std; 16 using namespace Geometry; 17 double radius, length, width, base, height; 18 cout << "Enter radius of the circle: "; 19 cin >> radius; 20 cout << "Area of the circle: " << area(radius) << endl; 21 cout << "Enter length and width of the rectangle: "; 22 cin >> length >> width; 23 cout << "Area of the rectangle: " << area(length, width) << endl; 24 cout << "Enter base and height of the triangle: "; 25 cin >> base >> height; 26 cout << "Area of the triangle: " << area(base, height, true) << endl; 27 return 0; 28 }</pre>	<pre>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 ===</pre>

Problem 3 :

main.cpp

```
1  #include <iostream>
2  using namespace std;
3  class Matrix {
4  public:
5      int rows, cols;
6      int **data;
7      Matrix(int r, int c) : rows(r), cols(c) {
8          data = new int*[rows];
9          for(int i = 0; i < rows; ++i) {
10             data[i] = new int[cols];
11         }
12     }
13     ~Matrix() {
14         for(int i = 0; i < rows; ++i) {
15             delete[] data[i];
16         }
17         delete[] data;
18     }
19     void input() {
20         cout << "Enter elements of the matrix (" << rows << " x " << cols << "
21             << "):\n";
22         for (int i = 0; i < rows; ++i) {
23             for (int j = 0; j < cols; ++j) {
24                 cin >> data[i][j];
25             }
26         }
```

main.cpp

```
27     void display() const {
28         for (int i = 0; i < rows; ++i) {
29             for (int j = 0; j < cols; ++j) {
30                 cout << data[i][j] << " ";
31             }
32             cout << endl;
33         }
34     }
35     Matrix add(const Matrix& mat) {
36         if (rows != mat.rows || cols != mat.cols) {
37             throw invalid_argument("Matrix dimensions must match for addition
38                 .");
39         }
40         Matrix result(rows, cols);
41         for (int i = 0; i < rows; ++i) {
42             for (int j = 0; j < cols; ++j) {
43                 result.data[i][j] = data[i][j] + mat.data[i][j];
44             }
45         }
46         return result;
47     }
48     Matrix multiply(const Matrix& mat) {
49         if (cols != mat.rows) {
50             throw invalid_argument("Number of columns of the first matrix must
51                 equal number of rows of the second matrix for multiplication."
52                 );
53         }
54     }
```

main.cpp



Share

Run

```
50     }
51     Matrix result(rows, mat.cols);
52     for (int i = 0; i < rows; ++i) {
53         for (int j = 0; j < mat.cols; ++j) {
54             result.data[i][j] = 0;
55             for (int k = 0; k < cols; ++k) {
56                 result.data[i][j] += data[i][k] * mat.data[k][j];
57             }
58         }
59     }
60     return result;
61 }
62 };
63 int main() {
64     int rows1, cols1, rows2, cols2;
65     cout << "Enter rows and columns of first matrix: ";
66     cin >> rows1 >> cols1;
67     Matrix mat1(rows1, cols1);
68     mat1.input();
69     cout << "Enter rows and columns of second matrix: ";
70     cin >> rows2 >> cols2;
71     Matrix mat2(rows2, cols2);
72     mat2.input();
73     try {
74         if (rows1 == rows2 && cols1 == cols2) {
75             Matrix sum = mat1.add(mat2);
76             cout << "Matrix Addition Result:\n";
77             sum.display();
78         } else {
79             cout << "Matrix dimensions do not match for addition." << endl;
80         }
81         if (cols1 == rows2) {
82             Matrix product = mat1.multiply(mat2);
83             cout << "Matrix Multiplication Result:\n";
84             product.display();
85         } else {
86             cout << "Matrix dimensions do not match for multiplication." << endl;
87         }
88     } catch (const exception& e) {
89         cout << "Error: " << e.what() << endl;
90     }
91     return 0;
92 }
```

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

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  class Shape {
4  public:
5      virtual double getArea() = 0;
6  };
7  class Rectangle : public Shape {
8  private:
9      double length;
10     double breadth;
11 public:
12     Rectangle(double l, double b) : length(l), breadth(b) {}
13     double getArea() override {
14         return length * breadth;
15     }
16 };
17 class Circle : public Shape {
18 private:
19     double radius;
20 public:
21     Circle(double r) : radius(r) {}
22     double getArea() override {
23         return M_PI * radius * radius;
24     }
25 };
```

```
26 class Triangle : public Shape {
27 private:
28     double base;
29     double height;
30 public:
31     Triangle(double b, double h) : base(b), height(h) {}
32     double getArea() override {
33         return 0.5 * base * height;
34     }
35 };
36 int main() {
37     Shape* shape1 = new Rectangle(5.0, 3.0);
38     Shape* shape2 = new Circle(4.0);
39     Shape* shape3 = new Triangle(6.0, 4.0);
40     cout << "Area of Rectangle: " << shape1->getArea() << endl;
41     cout << "Area of Circle: " << shape2->getArea() << endl;
42     cout << "Area of Triangle: " << shape3->getArea() << endl;
43     delete shape1;
44     delete shape2;
45     delete shape3;
46     return 0;
47 }
```


Output

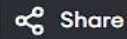
Area of Rectangle: 15
Area of Circle: 50.2655
Area of Triangle: 12

Problem 5 :

main.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  class Book {
4  protected:
5      string title;
6      string author;
7      string ISBN;
8      Book(string t, string a, string isbn) : title(t), author(a), ISBN(isbn) {}
9  void displayBookInfo() {
10      cout << "Book Title: " << title << endl;
11      cout << "Author: " << author << endl;
12      cout << "ISBN: " << ISBN << endl;
13  }
14 };
15 class Borrower {
16 protected:
17     string name;
18     int id;
19     string borrowedBook;
20 public:
21     Borrower(string n, int i) : name(n), id(i), borrowedBook("") {}
22 void displayBorrowerInfo() {
23     cout << "Borrower Name: " << name << endl;
24     cout << "Borrower ID: " << id << endl;
25     if (!borrowedBook.empty()) {
26         cout << "Borrowed Book: " << borrowedBook << endl;
27     } else {
28         cout << "No book borrowed." << endl;
29     }
30 }
```

main.cpp



Run

```
31- void borrowBook(string bookTitle) {
32-     borrowedBook = bookTitle;
33-     cout << name << " has borrowed the book: " << bookTitle << endl;
34- }
35- void returnBook() {
36-     if (!borrowedBook.empty()) {
37-         cout << name << " has returned the book: " << borrowedBook << endl;
38-         ;
39-         borrowedBook = "";
40-     } else {
41-         cout << name << " has no borrowed books to return." << endl;
42-     }
43- };
44- class Library : public Book, public Borrower {
45- public:
46-     Library(string bookTitle, string bookAuthor, string bookISBN, string
         borrowerName, int borrowerId)
47-         : Book(bookTitle, bookAuthor, bookISBN), Borrower(borrowerName,
         borrowerId) {}
48- void displayLibraryInfo() {
49-     displayBookInfo();
50-     displayBorrowerInfo();
51- }
52- };
```

```
53- int main() {
54-     Library library1("The Great Gatsby", "F. Scott Fitzgerald",
        "9780743273565", "John Doe", 101);
55-     library1.displayLibraryInfo();
56-     library1.borrowBook("The Great Gatsby");
57-     library1.displayLibraryInfo();
58-     library1.returnBook();
59-     library1.displayLibraryInfo();
60-     return 0;
61- }
```

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.