MID TERM LAB TASKS

REPORT

NAME **M.Ismail**

REG NO **SU-24-01-001-054**

SECTION **CS & CYB**

SEMISTER **2nd**

DATE **27/08/2025**

SUBMITTED TO **MR NIAMAT ULLAH SEB**

Task#1:

Implement a program to represent a simple calculator using classes and objects in C++. The calculator should be able to perform basic arithmetic operations such as addition, subtraction, multiplication, and division.

PROGRAM:

Task#01.M.Ismail(054).cpp

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

class Calculator {
public:
    float add(float num1, float num2) {
        return num1 + num2;
    }

    float subtract(float num1, float num2) {
        return num1 - num2;
    }

    float multiply(float num1, float num2) {
        return num1 * num2;
    }

    float divide(float num1, float num2) {
        if (num2 == 0) {
            cout << "Error: Division by zero is not allowed!" << endl;
            return 0;
        }
        return num1 / num2;
    }
};


int main() {
    Calculator calc;
    int choice;
    float num1, num2;

    cout << "Simple Calculator using Classes and Objects\n";
    cout << "-------------------------------------------\n";
    cout << "Select an operation:\n";
    cout << "1. Addition\n";
    cout << "2. Subtraction\n";
    cout << "3. Multiplication\n";
    cout << "4. Division\n";
    cout << "Enter your choice (1-4): ";
    cin >> choice;

    cout << "Enter first number: ";
    cin >> num1;
    cout << "Enter second number: ";
    cin >> num2;

    float result;

    switch (choice) {
        case 1:
            result = calc.add(num1, num2);
            cout << "Result: " << result << endl;
            break;
        case 2:
            result = calc.subtract(num1, num2);
            cout << "Result: " << result << endl;
            break;
        case 3:
            result = calc.multiply(num1, num2);
            cout << "Result: " << result << endl;
            break;
        case 4:
            result = calc.divide(num1, num2);
            if (num2 != 0) {
                cout << "Result: " << result << endl;
            }
            break;
        default:
            cout << "Invalid choice!" << endl;
    }

    return 0;
}
```

OUTPUT:

```
Simple Calculator using Classes and Objects
----------------------------------------
Select an operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice (1-4): 1
Enter first number: 10
Enter second number: 20
Result: 30

--------------------------------
```

==========End Task 1==========

Task #2:

Create a program to manage student information using classes and objects in C++. Each student should have attributes for their name, ID, and GPA. The program should allow users to create student objects, set their attributes, and display their information

PROGRAM:

Task#02.M.Ismail(054).cpp

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

class Student {
private:
    string name;
    int ID;
    float GPA;

public:
    void setName(string newName) {
        name = newName;
    }

    void setID(int newID) {
        ID = newID;
    }

    void setGPA(float newGPA) {
        GPA = newGPA;
    }

    string getName() {
        return name;
    }

    int getID() {
        return ID;
    }

    float getGPA() {
        return GPA;
    }
};

int main() {
    int numberOfStudents;
```

```
    cout << "Enter number of students: ";
    cin >> numberOfStudents;

    Student students[numberOfStudents];

    for (int i = 0; i < numberOfStudents; i++) {
        string name;
        int id;
        float gpa;

        cout << "\nEnter details for student " << (i + 1) << ":\n";

        cout << "Name: ";
        cin.ignore();
        getline(cin, name);

        cout << "ID: ";
        cin >> id;

        cout << "GPA: ";
        cin >> gpa;

        students[i].setName(name);
        students[i].setID(id);
        students[i].setGPA(gpa);
    }

    cout << "\n--- Student Information ---\n";
    for (int i = 0; i < numberOfStudents; i++) {
        cout << "Student " << (i + 1) << ":\n";
        cout << "Name: " << students[i].getName() << endl;
        cout << "ID: " << students[i].getID() << endl;
        cout << "GPA: " << students[i].getGPA() << endl;
        cout << "-----------------------------\n";
    }

    return 0;
}
```

OUTPUT:

```
Enter number of students: 1

Enter details for student 1:
Name: Irfan khan
ID: 053
GPA: 3.38

--- Student Information ---
Student 1:
Name: Irfan khan
ID: 53
GPA: 3.38
-----------------------------
```

==========End Task 02==========

Task #3:

Student Record Management System

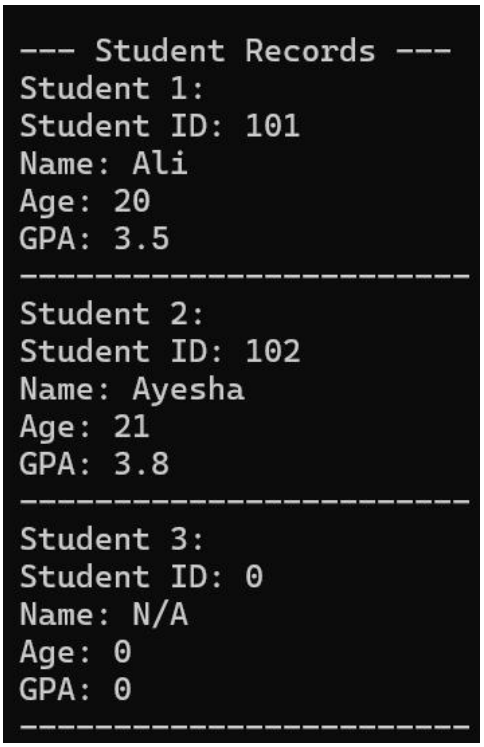
Problem Statement: Develop a program to manage student records efficiently. The system should allow **adding**, **displaying**, and **updating** information for each student.

PROGRAM:

Task#03.M.Ismail(054).cpp

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

class Student {
private:
    int StudentID;
    string Name;
    int Age;
    double GPA;

public:
    // Default constructor
    Student() {
        StudentID = 0;
        Name = "N/A";
        Age = 0;
        GPA = 0.0;
    }

    // Parameterized constructor
    Student(int id, string name, int age, double gpa) {
        StudentID = id;
        Name = name;
        Age = age;
        GPA = gpa;
    }

    // Display student info
    void Display() {
        cout << "Student ID: " << StudentID << endl;
        cout << "Name: " << Name << endl;
        cout << "Age: " << Age << endl;
        cout << "GPA: " << GPA << endl;
        cout << "-------------------------" << endl;
    }

    // Update student info
    void Update() {
        cout << "Enter new Student ID: ";
        cin >> StudentID;
        cin.ignore(); // Clear newline
        cout << "Enter new Name: ";
        getline(cin, Name);
        cout << "Enter new Age: ";
        cin >> Age;
        cout << "Enter new GPA: ";
        cin >> GPA;
    }
};

int main() {
    const int size = 3;
    Student students[size];

    //parameterized constructor
    students[0] = Student(101, "Ali", 20, 3.5);
    students[1] = Student(102, "Ayesha", 21, 3.8);

    // Default constructor
    students[2] = Student();

    // Display
    cout << "\n--- Student Records ---\n";
    for (int i = 0; i < size; i++) {
        cout << "Student " << (i + 1) << ":\n";
        students[i].Display();
    }

    cout << "\nUpdating information for Student 3:\n";
    students[2].Update();

    cout << "\n--- Updated Student Records ---\n";
    for (int i = 0; i < size; i++) {
        cout << "Student " << (i + 1) << ":\n";
        students[i].Display();
    }

    return 0;
}
```

OUTPUT:



==========End Task 03==========

Task #4:

Library Management System

Problem Statement: Design and implement a library management system to efficiently manage book records. The system should provide functionalities to add, display, and update information for each book in the library.

PROGRAM:

Task#04.M.Ismail(054).cpp

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

class Book {
private:
    string ISBN;
    string Title;
    string Author;
    string Genre;

public:
    Book() {
        ISBN = "N/A";
        Title = "N/A";
        Author = "N/A";
        Genre = "N/A";
    }

    // Parameterized constructor
    Book(string isbn, string title, string author, string genre) {
        ISBN = isbn;
        Title = title;
        Author = author;
        Genre = genre;
    }
    void Display() {
        cout << "ISBN: " << ISBN << endl;
        cout << "Title: " << Title << endl;
        cout << "Author: " << Author << endl;
        cout << "Genre: " << Genre << endl;
        cout << "-------------------------" << endl;
    }
    void Update() {
        cout << "Enter new ISBN: ";
        getline(cin, ISBN);
        cout << "Enter new Title: ";
        getline(cin, Title);
        cout << "Enter new Author: ";
        getline(cin, Author);
        cout << "Enter new Genre: ";
        getline(cin, Genre);
    }
};
int main() {
    const int size = 3;
    Book books[size];
    books[0] = Book("978-1234567890", "C++ Programming", "Bjarne Stroustrup", "Programming");
    books[1] = Book("978-9876543210", "Clean Code", "Robert C. Martin", "Software Engineering");

    books[2] = Book();
    cout << "\n--- Book Records ---\n";
    for (int i = 0; i < size; i++) {
        cout << "Book " << (i + 1) << ":\n";
        books[i].Display();
    }

    cout << "\nUpdating information for Book 3:\n";
    cin.ignore();
    books[2].Update();

    cout << "\n--- Updated Book Records ---\n";
    for (int i = 0; i < size; i++) {
        cout << "Book " << (i + 1) << ":\n";
        books[i].Display();
    }

    return 0;
}
```

OUTPUT:

```
--- Book Records ---
Book 1:
ISBN: 978-1234567890
Title: C++ Programming
Author: Bjarne Stroustrup
Genre: Programming
----------------------
Book 2:
ISBN: 978-9876543210
Title: Clean Code
Author: Robert C. Martin
Genre: Software Engineering
----------------------
Book 3:
ISBN: N/A
Title: N/A
Author: N/A
Genre: N/A
----------------------

Updating information for Book 3:
OOP
Enter new ISBN: Enter new Title: 173-908478
Enter new Author: Irfan khan
Enter new Genre: cyber security
```

```
--- Updated Book Records ---
Book 1:
ISBN: 978-1234567890
Title: C++ Programming
Author: Bjarne Stroustrup
Genre: Programming
--------------------------
Book 2:
ISBN: 978-9876543210
Title: Clean Code
Author: Robert C. Martin
Genre: Software Engineering
--------------------------
Book 3:
ISBN: OP
Title: 173-908478
Author: Irfan khan
Genre: cyber security
--------------------------
```

==========End Task 04==========

Lab Task #5:

Student Class Implementation

PROGRAM:

Task#05.M.Ismail(054).cpp

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

class Student {
private:
    string name;
    int rollNumber;
    char grade;

public:
    // Default constructor
    Student() {
        name = "Unknown";
        rollNumber = 0;
        grade = 'F';
    }

    // Parameterized constructor
    Student(string n, int r, char g) {
        name = n;
        rollNumber = r;
        grade = g;
    }

    void display() {
        cout << "Name: " << name << endl;
        cout << "Roll Number: " << rollNumber << endl
        cout << "Grade: " << grade << endl;
    }
};

int main() {
    Student student1;

    Student student2("Niamat", 101, 'A');

    cout << "Student 1:" << endl;
    student1.display();

    cout << "\nStudent 2:" << endl;
    student2.display();

    return 0;
}
```

OUTPUT

```
Student 1:
Name: Unknown
Roll Number: 0
Grade: F

Student 2:
Name: Niamat
Roll Number: 101
Grade: A

------------------
```

==========End Task 05==========

Task #6:

Implementing a Product Class with Overloaded Constructors

PROGRAM:

Task#06.M.Ismail(054).cpp

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

class Product {
private:
    string name;
    string id;
    double price;
    int quantity;
    string category;

public:
    Product() {
        name = "Unknown";
        id = "0000";
        price = 0.0;
        quantity = 0;
        category = "Unknown";
    }
    Product(string n, string i, double p, int q, string c)
        name = n;
        id = i;
        price = p;
        quantity = q;
        category = c;
    }
    Product(string n, string i, double p, string c) {
        name = n;
        id = i;
        price = p;
        quantity = 0;
        category = c;
    }
    Product(string n, string i, double p, int q) {
        name = n;
        id = i;
        price = p;
        quantity = q;
        category = "Unknown";
    }
    Product(string n, double p, int q, string c) {
        name = n;
        id = "0000";
        price = p;
        quantity = q;
        category = c;
    }
    Product(string i, double p, int q, string c,int r) {
        name = "Unknown";
        id = i;
        price = p;
        quantity = q;
        category = c;
        int price = r;
    }
    void display() {
        cout << fixed << setprecision(2);
        cout << "Name: " << name << endl;
        cout << "ID: " << id << endl;
        cout << "Price: $" << price << endl;
        cout << "Quantity: " << quantity << endl;
        cout << "Category: " << category << endl;
    }
};

int main() {
    Product product1;  // Default constructor
    Product product2("Laptop", "LAP123", 799.99, 10, "Electronics");
    Product product3("iPhone", "IPH456", 999.99, "Electronics");
    Product product4("PRO987", 19.99, 30, "Books");
    Product product5("PRO567", 5.99, 50, "Stationery");

    cout << "Product 1:\n";
    product1.display();
    cout << "\nProduct 2:\n";
    product2.display();
    cout << "\nProduct 3:\n";
    product3.display();
    cout << "\nProduct 4:\n";
    product4.display();
    cout << "\nProduct 5:\n";
    product5.display();

    return 0;
}
```

OUTPUT

```
Product 1:
Name: Unknown
ID: 0000
Price: $0.00
Quantity: 0
Category: Unknown

Product 2:
Name: Laptop
ID: LAP123
Price: $799.99
Quantity: 10
Category: Electronics

Product 3:
Name: iPhone
ID: IPH456
Price: $999.99
Quantity: 0
Category: Electronics

Product 4:
Name: PRO987
ID: 0000
Price: $19.99
Quantity: 30
Category: Books

Product 5:
Name: PRO567
ID: 0000
Price: $5.99
Quantity: 50
Category: Stationery

-------------------------
```

==========End Task 06==========

Task #7:

Implementing a Simple Counter Class with

Constructors and Destructor

PROGRAM:

Task#07.M.Ismail(054).cpp

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

class Counter {
private:
    int count;

public:
    Counter() {
        count = 0;
        cout << "Counter initialized with 0 (default constructor)." << endl;
    }
    Counter(int initialCount) {
        count = initialCount;
        cout << "Counter initialized with " << count << " (parameterized constructor)." << endl;
    }
    ~Counter() {
        cout << "Destructor called. Final count value was: " << count << endl;
    }
    void increment() {
        count++;
    }
    void display() {
        cout << "Current count: " << count << endl;
    }
};

int main() {
    Counter counter1;

    Counter counter2(5);

    counter1.increment();
    counter2.increment();
    cout << "\nCounter 1:" << endl;
    counter1.display();

    cout << "\nCounter 2:" << endl;
    counter2.display();

    return 0;
}
```

OUTPUT:

```
Counter initialized with 0 (default constructor).
Counter initialized with 5 (parameterized constructor).

Counter 1:
Current count: 1

Counter 2:
Current count: 6
Destructor called. Final count value was: 6
Destructor called. Final count value was: 1

--------------------------------
```

==========End Task 07==========

TASK#8: Library Management

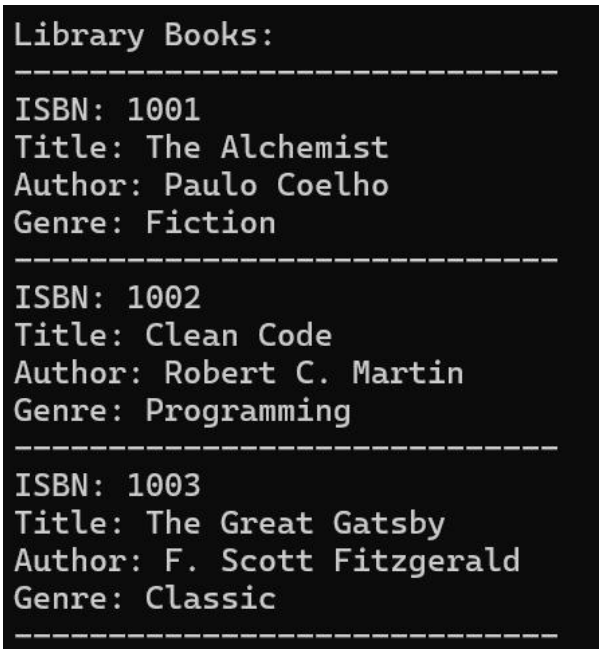
Using: Static variables, class implementation for book management with unique ISBN assignment, registration, and display

PROGRAM:

Task#08.M.Ismail(054).cpp

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

class Book {
private:
    static int isbnCounter;
    int isbn;
    string title;
    string author;
    string genre;

public:
    Book(string t, string a, string g) {
        isbn = ++isbnCounter;
        title = t;
        author = a;
        genre = g;
    }

    void display() const {
        cout << "ISBN: " << isbn << endl;
        cout << "Title: " << title << endl;
        cout << "Author: " << author << endl;
        cout << "Genre: " << genre << endl;
        cout << "------------------------------" << endl;
    }
};

int Book::isbnCounter = 1000;

int main() {
    vector<Book> library;

    library.push_back(Book("The Alchemist", "Paulo Coelho", "Fiction"));
    library.push_back(Book("Clean Code", "Robert C. Martin", "Programming"));
    library.push_back(Book("The Great Gatsby", "F. Scott Fitzgerald", "Classic"));

    cout << "\nLibrary Books:\n";
    cout << "-----------------------------\n";
    for (const Book& b : library) {
        b.display();
    }

    return 0;
}
```

OUTPUT:



==========End Task 08==========

TASK#9:

StringUtils

Using: Static member function for vowel counting, demonstrating its usage in C++.

PROGRAM:

Task#09.M.Ismail(054).cpp

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

class StringUtils {
public:
    static int countVowels(const string& str) {
        int count = 0;
        for (char ch : str) {
            char c = tolower(ch);
            if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {
                count++;
            }
        }
        return count;
    }
};
int main() {
    string input;
    cout << "Enter a string: ";
    getline(cin, input);
    int vowelCount = StringUtils::countVowels(input);
    cout << "Number of vowels: " << vowelCount << endl;

    return 0;
}
```

OUTPUT:

```
Enter a string: OOP
Number of vowels: 2

--------------------------------
```

==========End Task 09==========

TASK#10:

1. Arithmetic Operations
2. Using: Friend functions, classes for managing two numbers with
3. arithmetic operations, user input, and result display.

PROGRAM :

Task#10.M.Ismail(054).cpp

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

class Arithmetic {
private:
    float num1, num2;

public:
    void input() {
        cout << "Enter first number: ";
        cin >> num1;
        cout << "Enter second number: ";
        cin >> num2;
    }

    friend void add(const Arithmetic&);
    friend void subtract(const Arithmetic&);
    friend void multiply(const Arithmetic&);
    friend void divide(const Arithmetic&);
};
void add(const Arithmetic& obj) {
    cout << "Addition: " << obj.num1 + obj.num2 << endl;
}
void subtract(const Arithmetic& obj) {
    cout << "Subtraction: " << obj.num1 - obj.num2 << endl;
}

void multiply(const Arithmetic& obj) {
    cout << "Multiplication: " << obj.num1 * obj.num2 << endl;
}

void divide(const Arithmetic& obj) {
    if (obj.num2 != 0)
        cout << "Division: " << obj.num1 / obj.num2 << endl;
    else
        cout << "Division by zero is not allowed!" << endl;
}
int main() {
    Arithmetic calc;

    calc.input();

    cout << "\n--- Arithmetic Operations ---" << endl;
    add(calc);
    subtract(calc);
    multiply(calc);
    divide(calc);

    return 0;
}
```

OUTPUT:

```
Enter first number: 10
Enter second number: 5

--- Arithmetic Operations ---
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2

--------------------------------
```

==========End Task 10==========

Lab Task 11: Develop a C++ program to manage a library system. The program should

allow users to register new books in the library catalog. Each book should be assigned a unique ISBN (International Standard Book Number). Implement a class Book to represent book objects, ensuring that each book has a unique ISBN. After registering three books, display the details of each book, including its title, author, and ISBN.

PROGRAM:

Task#11.M.Ismail(054).cpp

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

class Book {
private:
    static int nextISBN;
    int ISBN;
    string title;
    string author;

public:
    // Constructor
    Book(string t, string a) {
        ISBN = nextISBN++;
        title = t;
        author = a;
    }
    void setTitle(string t) { title = t; }
    void setAuthor(string a) { author = a; }

    string getTitle() { return title; }
    string getAuthor() { return author; }
    int getISBN() { return ISBN; }

    // Display book details
    void displayDetails() const {
        cout << "Title: " << title << endl;
        cout << "Author: " << author << endl;
        cout << "ISBN: " << ISBN << endl;
    }
};

int Book::nextISBN = 1001;

int main() {
    cout << "Registering Books in Library Catalog..." << endl << endl;

    Book book1("The Great Gatsby", "F. Scott Fitzgerald");
    Book book2("To Kill a Mockingbird", "Harper Lee");
    Book book3("1984", "George Orwell");
    Book book4("The 7 Habits of Highly Effective People", "Stephen R. Covey");

    cout << "Book Details:\n-----------------\n";

    cout << "Book 1:\n";
    book1.displayDetails();
    cout << "\nBook 2:\n";
    book2.displayDetails();
    cout << "\nBook 3:\n";
    book3.displayDetails();
    cout << "\nBook 4 (Motivational):\n";
    book4.displayDetails();

    return 0;
}
```

OUTPUT:

```
Registering Books in Library Catalog...

Book Details:
----------------
Book 1:
Title: The Great Gatsby
Author: F. Scott Fitzgerald
ISBN: 1001

Book 2:
Title: To Kill a Mockingbird
Author: Harper Lee
ISBN: 1002

Book 3:
Title: 1984
Author: George Orwell
ISBN: 1003

Book 4 (Motivational):
Title: The 7 Habits of Highly Effective People
Author: Stephen R. Covey
ISBN: 1004

--------------------------------
```

==========End Task 11==========

Lab Task #12 : You are tasked with creating a class StringUtils to perform operations on

strings. Implement a static member function to count the number of vowels in a given string. Write a C++ program to demonstrate the usage of this static member function.

PROGRAM:

Task#12.M.Ismail(054).cpp

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

class StringUtils {
public:
    static int countVowels(const string& str) {
        int count = 0;
        for (char ch : str) {
            char lowerCh = tolower(ch); // Convert to lowercase for uniform comparison
            if (lowerCh == 'a' || lowerCh == 'e' || lowerCh == 'i' ||
                lowerCh == 'o' || lowerCh == 'u') {
                count++;
            }
        }
        return count;
    }
};

int main() {
    string input;

    cout << "Enter a string: ";
    getline(cin, input);

    int vowelCount = StringUtils::countVowels(input);

    cout << "Number of vowels in the string: " << vowelCount << endl;

    return 0;
}
```

OUTPUT:

```
Enter a string: M.Ismail
Number of vowels in the string: 2

--------------------------------
```

==========End Task 12==========

Lab Task 13:

Create a program to manage two numbers and perform basic arithmetic operations on them. There are two classes involved: **Number** and **Calculator**.

Program:

Task#13.M.Ismail(054).cpp

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

class Calculator;

class Number {
private:
    double num1, num2;

public:
    void setNumber1(double n1) {
        num1 = n1;
    }

    void setNumber2(double n2) {
        num2 = n2;
    }

    double getNumber1() const {
        return num1;
    }

    double getNumber2() const {
        return num2;
    }
    friend void calculate(Number& n);
};

class Calculator {
public:
    double add(double a, double b) {
        return a + b;
    }

    double subtract(double a, double b) {
        return a - b;
    }

    double multiply(double a, double b) {
        return a * b;
    }
```

```
    double divide(double a, double b) {
        if (b == 0) {
            cout << "Error: Division by zero!" << endl;
            return 0;
        }
        return a / b;
    }
};

void calculate(Number& n) {
    Calculator calc;
    int choice;
    string again;

    cout << "Welcome to the Basic Arithmetic Calculator!" << endl;

    do {
        cout << "\nArithmetic Operations:" << endl;
        cout << "1. Addition" << endl;
        cout << "2. Subtraction" << endl;
        cout << "3. Multiplication" << endl;
        cout << "4. Division" << endl;
        cout << "Enter your choice (1-4): ";
        cin >> choice;

        switch (choice) {
        case 1:
            cout << "Result of addition: " << n.num1 << " + " << n.num2 << " = "
                 << calc.add(n.num1, n.num2) << endl;
            break;
        case 2:
            cout << "Result of subtraction: " << n.num1 << " - " << n.num2 << " = "
                 << calc.subtract(n.num1, n.num2) << endl;
            break;
        case 3:
            cout << "Result of multiplication: " << n.num1 << " * " << n.num2 << " = "
                 << calc.multiply(n.num1, n.num2) << endl;
            break;
        case 4:
            cout << "Result of division: " << n.num1 << " / " << n.num2 << " = "
                 << calc.divide(n.num1, n.num2) << endl;
            break;
        default:
            cout << "Invalid choice. Please enter a number between 1 and 4." << endl;
        }

        cout << "Do you want to perform another operation? (yes/no): ";
        cin >> again;
    } while (again == "yes" || again == "Yes");

    cout << "Thank you for using the Basic Arithmetic Calculator!" << endl;
}

int main() {
    Number n;
    double a, b;

    cout << "Enter the first number: ";
    cin >> a;
    cout << "Enter the second number: ";
    cin >> b;

    n.setNumber1(a);
    n.setNumber2(b);

    calculate(n);

    return 0;
}
```

OUTPUT:

```
Enter the first number: 13
Enter the second number: 3
Welcome to the Basic Arithmetic Calculator!

Arithmetic Operations:
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice (1-4): 1
Result of addition: 13 + 3 = 16
Do you want to perform another operation? (yes/no): no
Thank you for using the Basic Arithmetic Calculator!

--------------------------------
```

==========End Task 13==========

Lab Task 14:

: Design a C++ program to simulate a smartwatch fitness tracker. Implement a class named StepCounter to

monitor the number of steps a person takes daily. Overload the increment (++) and decrement (--) operators in both prefix and postfix

forms to accurately reflect the changes in step count throughout the day. Ensure that the overloaded operators maintain the integrity of

the step count data and provide appropriate feedback on the progress of the user's activity.

Program:

Task#14.M.Ismail(054).cpp

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

class StepCounter {
private:
    int stepCount;

public:
    // Constructor
    StepCounter() : stepCount(0) {}

    // Prefix increment (++step)
    StepCounter& operator++() {
        ++stepCount;
        return *this;
    }

    // Postfix increment (step++)
    StepCounter operator++(int) {
        StepCounter temp = *this;
        stepCount++;
        return temp;
    }

    // Prefix decrement (--step)
    StepCounter& operator--() {
        if (stepCount > 0)
            --stepCount;
        else
            cout << "Step count can't go below 0.\n";
        return *this;
    }

    // Postfix decrement (step--)
    StepCounter operator--(int) {
        StepCounter temp = *this;
        if (stepCount > 0)
            stepCount--;
        else
            cout << "Step count can't go below 0.\n";
        return temp;
    }

    // Display current step count
    void display() const {
        cout << "Current step count: " << stepCount << end
    }
};

int main() {
    StepCounter sc;

    cout << "Initial step count: ";
    sc.display();

    ++sc; // Prefix increment
    cout << "Step count after prefix increment: ";
    sc.display();

    sc++; // Postfix increment
    cout << "Step count after postfix increment: ";
    sc.display();

    --sc; // Prefix decrement
    cout << "Step count after prefix decrement: ";
    sc.display();

    sc--; // Postfix decrement
    cout << "Step count after postfix decrement: ";
    sc.display();

    return 0;
}
```

OUTPUT:

```
Initial step count: Current step count: 0
Step count after prefix increment: Current step count: 1
Step count after postfix increment: Current step count: 2
Step count after prefix decrement: Current step count: 1
Step count after postfix decrement: Current step count: 0

--------------------------------
```

==========End Task 14==========

Lab Task 15:

: Develop a C++ program for a simple social media platform. Implement a class named User to represent users

of the platform. Overload the unary increment (++) operator as a friend function to increase a user's follower count when another

user follows them. Similarly, overload the unary decrement (--) operator as a friend function to decrease a user's follower count when

another user unfollows them.

Program:

Task#15.M.Ismail(054).cpp

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

class User {
private:
    string username;
    int followerCount;

public:
    User(const string& name) {
        username = name;
        followerCount = 0;
    }

    void display() const {
        cout << "Username: " << username << ", Followers: " << followerCount << endl;
    }

    friend User& operator++(User& user) {
        user.followerCount++;
        return user;
    }

    friend User& operator--(User& user) {
        if (user.followerCount > 0)
            user.followerCount--;
        else
            cout << user.username << " has no followers to remove." << endl;
        return user;
    }
};

int main() {
    User user1("Alice");
    User user2("Bob");

    cout << "Initial follower counts:" << endl;
    user1.display();
    user2.display();

    cout << "\nBob follows Alice (++Alice)" << endl;
    ++user1;

    cout << "Alice follows Bob (++Bob)" << endl;
    ++user2;
    cout << "\nFollower counts after following:" << endl;
    user1.display();
    user2.display();

    cout << "\nBob unfollows Alice (--Alice)" << endl;
    --user1;

    cout << "\nFollower counts after unfollowing:" << endl;
    user1.display();
    user2.display();

    return 0;
}
```

OUTPUT:

==========End Task 15==========

Lab Task 16:

Imagine a scenario where you're developing a simulation for a smart home system. You have a class named

SmartDevice representing various smart devices in the home, such as lights, thermostats, and door locks. Implement the unary

increment (++) operator as a friend function to increase the brightness of a light when the operator is applied. Similarly, overload the

unary decrement (--) operator as a friend function to decrease the brightness of a light when the operator is applied

Program:

Task#16.M.Ismail(054).cpp

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

class SmartDevice {
private:
    string deviceName;
    int brightnessLevel;

public:
    SmartDevice(const string& name) {
        deviceName = name;
        brightnessLevel = 0;
    }

    void display() const {
        cout << "Device: " << deviceName << ", Brightness: " << brightnessLevel << endl;
    }

    friend SmartDevice& operator++(SmartDevice& device) {
        device.brightnessLevel += 10;
        return device;
    }

    friend SmartDevice& operator--(SmartDevice& device) {
        if (device.brightnessLevel >= 10)
            device.brightnessLevel -= 10;
        else
            device.brightnessLevel = 0;
        return device;
    }
};

int main() {
    SmartDevice light1("Living Room Light");
    SmartDevice light2("Bedroom Light");

    cout << "Initial Brightness Levels:" << endl;
    light1.display();
    light2.display();

    ++light1;
    ++light1;
    ++light2;

    cout << "\nAfter Increasing Brightness:" << endl;
    light1.display();
    light2.display();

    --light1;
    --light2;

    cout << "\nAfter Decreasing Brightness:" << endl;
    light1.display();
    light2.display();

    return 0;
}
```

OUTPUT:

```
Initial Brightness Levels:
Device: Living Room Light, Brightness: 0
Device: Bedroom Light, Brightness: 0

After Increasing Brightness:
Device: Living Room Light, Brightness: 20
Device: Bedroom Light, Brightness: 10

After Decreasing Brightness:
Device: Living Room Light, Brightness: 10
Device: Bedroom Light, Brightness: 0

--------------------------------
```

==========End Task 16==========

Lab Task 17:

Design a C++ program to perform arithmetic operations on complex numbers. Complex numbers

consist of a real part and an imaginary part, denoted as a + bi, where a is the real part, b is the

imaginary part, and i is the imaginary unit. Implement a class named Complex to represent

complex numbers. The program should support addition, subtraction, multiplication, and division

of complex numbers using operator overloading. Additionally, implement unary operator overloading

to support negation and conjugation of complex numbers.

Program:

Task#17.M.Ismail(054).cpp

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

class Complex {
private:
    double real;
    double imaginary;

public:
    Complex(double r = 0, double i = 0) {
        real = r;
        imaginary = i;
    }

    double getReal() const {
        return real;
    }

    double getImaginary() const {
        return imaginary;
    }

    Complex operator+(const Complex& other) const {
        return Complex(real + other.real, imaginary + other.imaginary);
    }

    Complex operator-(const Complex& other) const {
        return Complex(real - other.real, imaginary - other.imaginary);
    }

    Complex operator*(const Complex& other) const {
        double r = real * other.real - imaginary * other.imaginary;
        double i = real * other.imaginary + imaginary * other.real;
        return Complex(r, i);
    }

    Complex operator/(const Complex& other) const {
        double denominator = other.real * other.real + other.imaginary * other.imaginary;
        double r = (real * other.real + imaginary * other.imaginary) / denominator;
        double i = (imaginary * other.real - real * other.imaginary) / denominator;
        return Complex(r, i);
    }

    Complex operator-() const {
        return Complex(-real, -imaginary);
    }

    Complex operator~() const {
        return Complex(real, -imaginary);
    }

    void display() const {
        cout << real;
        if (imaginary >= 0)
            cout << " + " << imaginary << "i";
        else
            cout << " - " << -imaginary << "i";
    }
};

int main() {
    Complex c1, c2;
    double r, i;
    int choice;

    cout << "Complex Number Calculator\n";
    cout << "Enter the real and imaginary parts of the first complex number:\n";
    cout << "Real part: ";
    cin >> r;
    cout << "Imaginary part: ";
    cin >> i;
    c1 = Complex(r, i);

    cout << "Enter the real and imaginary parts of the second complex number:\n";
    cout << "Real part: ";
    cin >> r;
    cout << "Imaginary part: ";
    cin >> i;
    c2 = Complex(r, i);

    do {
        cout << "\n1. Addition\n2. Subtraction\n3. Multiplication\n4. Division\n5. Negation\n6. Conjugate\n7. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
```

```
        switch (choice) {
        case 1: {
            Complex result = c1 + c2;
            cout << "Result of addition: ";
            c1.display();
            cout << " + ";
            c2.display();
            cout << " = ";
            result.display();
            cout << endl;
            break;
        }
        case 2: {
            Complex result = c1 - c2;
            cout << "Result of subtraction: ";
            c1.display();
            cout << " - ";
            c2.display();
            cout << " = ";
            result.display();
            cout << endl;
            break;
        }
        case 3: {
            Complex result = c1 * c2;
            cout << "Result of multiplication: ";
            c1.display();
            cout << " * ";
            c2.display();
            cout << " = ";
            result.display();
            cout << endl;
            break;
        }
        case 4: {
            Complex result = c1 / c2;
            cout << "Result of division: ";
            c1.display();
            cout << " / ";
            c2.display();
            cout << " = ";
            result.display();
```

```
            cout << endl;
            break;
        }
        case 5: {
            Complex result = -c1;
            cout << "Result of negation: -";
            c1.display();
            cout << " = ";
            result.display();
            cout << endl;
            break;
        }
        case 6: {
            Complex result = ~c1;
            cout << "Result of conjugate: ~";
            c1.display();
            cout << " = ";
            result.display();
            cout << endl;
            break;
        }
        case 7:
            cout << "Exiting the program...\n";
            break;
        default:
            cout << "Invalid choice. Try again.\n";
        }

    } while (choice != 7);

    return 0;
}
```

OUTPUT:

```
Complex Number Calculator
Enter the real and imaginary parts of the first complex number:
Real part: 12
Imaginary part: 6
Enter the real and imaginary parts of the second complex number:
Real part: 21
Imaginary part: 7

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Negation
6. Conjugate
7. Exit
Enter your choice: 1
Result of addition: 12 + 6i + 21 + 7i = 33 + 13i

1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Negation
6. Conjugate
7. Exit
Enter your choice: 7
Exiting the program...

--------------------------------
```

==========End Task 17==========

Lab Task 18:

Problem Description:

Your task is to implement a class named Date to represent dates with day, month, and year components. The class

should have the following specifications:

Class Definition (Date):

•

Declare a class named Date with three private members: day, month, and year.

•

Provide a constructor that initializes the day, month, and year with the provided values. If no values are

provided, default to the 1st day of January 2000.

Overloaded Operators:

•

operator>>: Overload the extraction operator (>>). This operator should prompt the user to enter the day,

month, and year separately and assign them to the corresponding private members of the Date object. It

should return the input stream for chaining.

•

operator<<: Overload the insertion operator (<<). This operator should output the date in the format

"day/month/year" to the output stream. It should return the output stream for chaining.

Main Function:

•

Create an instance of the Date class named date.

•

Prompt the user to enter a date in the format "day month year".

•

Use the overloaded operator>> to input data into date.

•

Use the overloaded operator<< to output the date entered.

Ensure that the program operates as described, handling user input and output correctly, and following good

programming practices and principles.

Output

Enter day, month, and year (format: dd mm yyyy): 25 12 2023

The date entered is: 25/12/2023

Program:

Task#18.M.Ismail(054).cpp

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

class Date {
private:
    int day, month, year;

public:
    Date(int d = 1, int m = 1, int y = 2000) {
        day = d;
        month = m;
        year = y;
    }

    friend istream& operator>>(istream& in, Date& date) {
        in >> date.day >> date.month >> date.year;
        return in;
    }

    friend ostream& operator<<(ostream& out, const Date& date) {
        out << date.day << '/' << date.month << '/' << date.year;
        return out;
    }
};

int main() {
    Date date;
    cout << "Enter day, month, and year (format: dd mm yyyy): ";
    cin >> date;
    cout << "The date entered is: " << date << endl;
    return 0;
}
```

OUTPUT:

```
Enter day, month, and year (format: dd mm yyyy): 1
11
2006
The date entered is: 1/11/2006

--------------------------------
```

==========End Task 18==========

Lab Task 19:

Develop a C++ program for managing inventory in a retail store. Create a class named Product to represent

individual items in the inventory. Overload the arithmetic operators (+, -, *, /) to support various operations such as combining

products, subtracting products, adjusting quantities, and calculating total prices. Additionally, implement compound assignment

operators (e.g., +=, -=) for efficient modification of product quantities. Overload the insertion (<<) and extraction (>>) operators to

simplify input and output operations when interacting with product data.

Program:

Task#19.M.Ismail(054).cpp

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

class Product {
private:
    string productName;
    double price;
    int quantity;

public:
    Product(string name = "", double p = 0.0, int q = 0) {
        productName = name;
        price = p;
        quantity = q;
    }

    Product operator+(const Product& other) const {
        return Product(productName + " + " + other.productName, price + other.price, quantity + other.quantity);
    }

    Product operator-(const Product& other) const {
        return Product(productName + " - " + other.productName, price - other.price, quantity - other.quantity);
    }

    Product operator*(int factor) const {
        return Product(productName + " * " + to_string(factor), price * factor, quantity);
    }

    Product operator/(int divisor) const {
        return Product(productName + " / " + to_string(divisor), price / divisor, quantity);
    }

    Product& operator+=(int qty) {
        quantity += qty;
        return *this;
    }

    Product& operator-=(int qty) {
        quantity -= qty;
        return *this;
    }

    double totalPrice() const {
        return price * quantity;
    }

    friend istream& operator>>(istream& in, Product& p) {
        cout << "Enter product name: ";
        in >> ws;
```

```
        getline(in, p.productName);
        cout << "Enter price: $";
        in >> p.price;
        cout << "Enter quantity: ";
        in >> p.quantity;
        return in;
    }

    friend ostream& operator<<(ostream& out, const Product& p) {
        out << "Product: " << p.productName << ", Price: $" << p.price << ", Quantity: " << p.quantity;
        return out;
    }
};

int main() {
    Product p1, p2;
    cout << "Enter details for product 1:\n";
    cin >> p1;
    cout << "Enter details for product 2:\n";
    cin >> p2;

    Product sum = p1 + p2;
    Product diff = p1 - p2;
    Product scaled = p1 * 2;
    Product divided = p2 / 3;

    cout << "Sum: " << sum << endl;
    cout << "Difference: " << diff << endl;
    cout << "Scaled Product: " << scaled << endl;
    cout << "Quotient: " << divided << endl;

    p1 += 10;
    p2 -= 15;

    cout << "Product 1 after adjustment: " << p1 << endl;
    cout << "Product 2 after adjustment: " << p2 << endl;

    cout << "Total price of product 1: $" << p1.totalPrice() << endl;
    cout << "Total price of product 2: $" << p2.totalPrice() << endl;

    return 0;
}
```

OUTPUT:

```
Enter details for product 1:
Enter product name: laptop
Enter price: $50,000
Enter quantity: Enter details for product 2:
Enter product name: Enter price: $Enter quantity: Sum: Product: laptop + , Price: $50, Quantity: 0
Difference: Product: laptop - , Price: $50, Quantity: 0
Scaled Product: Product: laptop * 2, Price: $100, Quantity: 0
Quotient: Product:  / 3, Price: $0, Quantity: 0
Product 1 after adjustment: Product: laptop, Price: $50, Quantity: 10
Product 2 after adjustment: Product: , Price: $0, Quantity: -15
Total price of product 1: $500
Total price of product 2: $-0

--------------------------------
```

==========End Task Mid Term Lab Report==========