Lab Report

NAME M.REHAN AFRIDI

REG NO. SU 23.01.001.036

DEP BS. COMPUTER SCIENCE

CLASS. SECTION. (B)

SUBJECT. OOP LAB

Lab 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.

Here's what your program should do:

1.Define a class called Calculator with the following attributes and

Methods:

- **Methods:**

- **add(float a, float b): Returns the sum of a and b.**

- **subtract(float a, float b): Returns the result of subtracting b**

From a.

- **multiply(float a, float b): Returns the product of a and b.**

- **divide(float a, float b): Returns the result of dividing a by b.**

In the main() function:

• Create an object of the Calculator class.

• Prompt the user to choose an operation (addition, subtraction,

Multiplication, or division).

• If the user chooses an operation, prompt for the two numbers to

Perform the operation on.

• Call the corresponding method of the Calculator object based on

The user's choice.

• Display the result of the operation

Program •

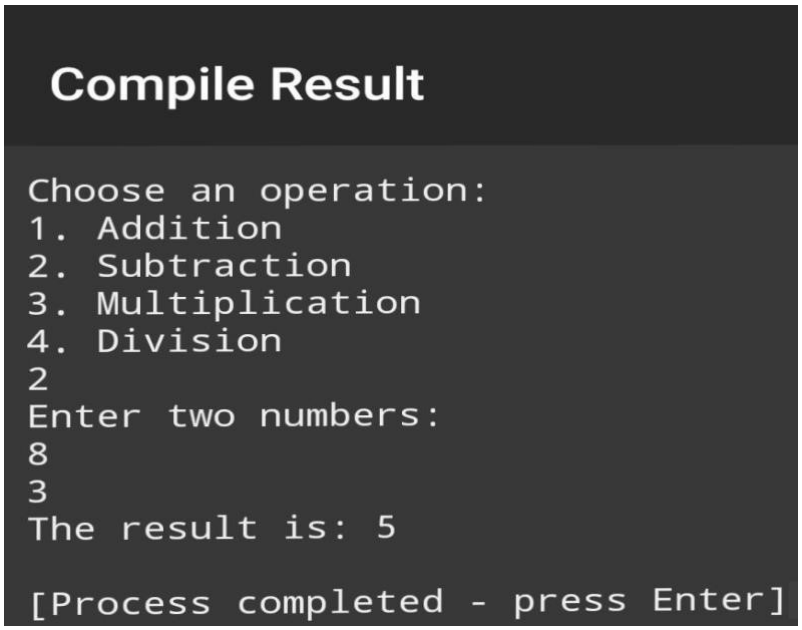
```
#include <iostream>
Using namespace std;
Class Calculator { Public:
   Float add(float a, float b) {
      Return a + b;
```

```
    }

    Float subtract(float a, float b) {
        Return a – b;
    }

    Float multiply(float a, float b) {
        Return a * b;
    }

    Float divide(float a, float b) {
        If (b != 0) {
            Return a / b;
        } else {
            Cout << “Error: Division by zero is not allowed.” << endl;
            Return 0;
        }
    }
};

Int main() {
    Calculator calculator;

    Int operation;
    Cout << “Choose an operation:” << endl;
    Cout << “1. Addition” << endl;
```

```
   Cout << “2. Subtraction” << endl;
   Cout << “3. Multiplication” << endl;
   Cout << “4. Division” << endl;
   Cin >> operation;

   Float a, b;
   Cout << “Enter two numbers:” << endl;
   Cin >> a >> b;

   Float result;
   Switch (operation)
{      Case 1:
         Result = calculator.add(a, b);
         Break;
Case 2:
         Result = calculator.subtract(a, b);
         Break;
Case 3:
         Result = calculator.multiply(a, b);
         Break;
Case 4:
         Result = calculator.divide(a, b);
         Break;
Default:
```

```
        Cout << “Invalid operation chosen.” <<endl;
        Return 1;}
    Cout << “The result is: “ << result << endl;

    Return 0;
    }
```

Out put



Lab 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.

Here's what your program should do:

1.Define a class called Student with the following attributes and

Methods:

- **Attributes:**

- **name (string): The name of the student.**

- **ID (integer): The ID of the student.**

- **GPA (float): The GPA of the student.**
- **Methods:**

- **setName(string newName): Sets the name of the student.**

- **setID(int newID): Sets the ID of the student.**

- **setGPA(float newGPA): Sets the GPA of the student.**

- **getName(): Returns the name of the student.**

- **getID(): Returns the ID of the student.**

- **getGPA(): Returns the GPA of the student.**

2.In the main() function:

- **Create an array or vector of Student objects to store multiple**

Student records.

- **Prompt the user to enter the name, ID, and GPA of each student**

And store this information in the Student objects.

- **Allow the user to input information for multiple students.**

- **Display the information of all the students entered.**

Program •

```
#include <iostream>

#include <vector>

#include <string>


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() const {
       Return name;
    }

    Int getID() const {
        Return ID;
    }
    Float getGPA() const {
        Return GPA;
    }
};

Int main() {
    Vector<Student> students;
    Char moreStudents = ‘y’;

    While (moreStudents == ‘y’ || moreStudents == ‘Y’) {
        Student newStudent;
```

```
        String name;
        Int id;
        Float gpa;

        Cout << “Enter the name of the student: “;
Getline(cin, name);         newStudent.setName(name);

         cout << “Enter the ID of the student: “;
cin >> id;         newStudent.setID(id);
        cout << “Enter the GPA of the student: “;
cin >> gpa;         newStudent.setGPA(gpa);

        students.push_back(newStudent);

        cout << “Do you want to add another student? (y/n): “;
cin >> moreStudents;         cin.ignore();
    }

    Cout << “\nDisplaying student information:\n”;
    For (const Student& student : students) {
        Cout << “Name: “ << student.getName() << “, ID: “ << student.getID() << “,
GPA: “ << student.getGPA() << endl;
```

```
    }

    Return 0;
}
```

Out put :

Compile Result

```
Enter the name of the student: jamal
Enter the ID of the student: 123
Enter the GPA of the student: 3.00
Do you want to add another student? (y/n)
: khan

Displaying student information:
Name: jamal, ID: 123, GPA: 3

[Process completed - press Enter]
```

Lab Tasks = 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.

Class: Student

Data Members:

- **StudentID (integer): Stores the unique identification number of the**

Student.

- **Name (string): Stores the name of the student.**

• Age (integer): Stores the age of the student.

• GPA (double): Stores the grade point average of the student.

Member Functions:

1.Default Constructor:

• Initializes the student's data members to default values.

Parameterized Constructor:

• Initializes the student's data members with provided values for

StudentID, name, age, and GPA.

3.Display():

• Displays the information of the student, including StudentID,

Name, age, and GPA.

4.Update():

• **Allows updating the information of the student by providing new**

Values for StudentID, name, age, and GPA.

Main Functionality:

1.Creates Student Objects:

• **Instantiates multiple Student objects using both default and**

Parameterized constructors to represent different students.

2.Displays Student Information:

• **Calls the display function for each Student object to show its**

Information, including StudentID, name, age, and GPA.

3.Updates Student Information:

• **Calls the update function to modify the information of selected**

Students, such as updating the StudentID, name, age, or GPA

Program..

```
#include <iostream>
#include <vector>
Using namespace std ;
Class Student { Public:
   Int StudentID;
   String Name;
   Int Age;
   Double GPA;
   Student() {
      StudentID = 0;
      Name = “Unknown”;
      Age = 0;
      GPA = 0.0;
   }
   Student(int id, string name, int age, double gpa) {       StudentID = id;
      Name = name;
      Age = age;
      GPA = gpa;
   }
```

```
    Void Display() const{
        Cout << “StudentID: “ << StudentID << endl;
        Cout << “Name: “ << Name << endl;
        Cout << “Age: “ << Age << endl;
        Cout << “GPA: “ << GPA << endl;
        Cout << endl;
    }
    Void Update(int id, string name, int age, double gpa) {
        StudentID = id;
        Name = name;
        Age = age;
        GPA = gpa;
    }
};

Int main() {
    Vector<Student> students;
    Student student1;
    Student student2(101, “John Doe”, 20, 3.5);
    Students.push_back(student1);
    Students.push_back(student2);
    Cout << “Student Information:” << endl;
    For (const Student& student : students) {
        Student.Display();
```

```
    }
    Students[0].Update(102, “Jane Doe”, 22, 3.8);
    Students[1].Update(101, “John Doe”, 21, 3.6);
    Cout << “Updated Student Information:” << endl;
    For (const Student& student : students) {
        Student.Display();
    }

    Return 0;
}
```

Out put..

Compile Result

```
Student Information:
StudentID: 0
Name: Unknown
Age: 0
GPA: 0

StudentID: 101
Name: John Doe
Age: 20
GPA: 3.5

Updated Student Information:
StudentID: 102
Name: Jane Doe
Age: 22
GPA: 3.8

StudentID: 101
Name: John Doe
Age: 21
GPA: 3.6


[Process completed - press Enter]
```

Lab Tasks= 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.

Class: Book

Data Members:

- **ISBN (string): Stores the unique International Standard Book Number**

Of the book.
- **Title (string): Stores the title of the book.**

- **Author (string): Stores the name of the author of the book.**

- **Genre (string): Stores the genre/category of the book.**

Member Functions:

1.Default Constructor:

- **Initializes the book's data members to default values.**

2.Parameterized Constructor:

- **Initializes the book's data members with provided values for ISBN,**

Title, author, and genre.

3.Display():

- **Displays the information of the book, including ISBN, title, author,**

And genre.

4.Update():

- **Allows updating the information of the book by providing new**

Values for ISBN, title, author, and genre.

Main Functionality:

1.Creates Book Objects:

- **Instantiates multiple Book objects using both default and**

Parameterized constructors to represent different books.

2.Displays Book Information:

- **Calls the display function for each Book object to show its**

Information, including ISBN, title, author, and genre.

3.Updates Book Information:

- **Calls the update function to modify the information of selected Books, such as updating the ISBN, title, author, or genre.**

Program..

```
#include <iostream>
#include <vector>
Using namespace std;
```

```
Class Book { Public:

   String ISBN;

   String Title;

   String Author;

   String Genre;

   Book() {

      ISBN = “Unknown”;

      Title = “Unknown”;

      Author = “Unknown”;

      Genre = “Unknown”;

   }

   Book(string isbn,string title,string author,string genre) {

      ISBN = isbn;
      Title = title;

      Author = author;

      Genre = genre;

   }



   Void Display()const {

      Cout << “ISBN: “ << ISBN << endl;

      Cout << “Title: “ << Title << endl;

      Cout << “Author: “ << Author << endl;
```

```
        Cout << “Genre: “ << Genre <<endl;
       Cout << endl;
    }
    Void Update(string isbn,string title, string author,string genre) {
        ISBN = isbn;
        Title = title;
        Author = author;
        Genre = genre;
    }
};

Int main() {
    Vector<Book> books;


    Book book1;
    Book book2(“1234567890”, “Book Title”, “Author Name”, “Fiction”);
    Books.push_back(book1);
    Books.push_back(book2);

    Cout << “Book Information:” <<endl;
    For (const Book& book : books) {
        Book.Display();
    }
```

```
    Books[0].Update("1111111111", "New Title", "New Author", "NonFiction");
        Books[1].Update("2222222222", "Updated Title", "Updated Author", "Fiction");
    Cout << "Updated Book Information:" <<endl;
    For (const Book& book : books) {
        Book.Display();
    }

    Return 0;
}
```

Out put..

Compile Result

```
Book Information:
ISBN: Unknown
Title: Unknown
Author: Unknown
Genre: Unknown

ISBN: 1234567890
Title: Book Title
Author: Author Name
Genre: Fiction

Updated Book Information:
ISBN: 1111111111
Title: New Title
Author: New Author
Genre: Non-Fiction

ISBN: 2222222222
Title: Updated Title
Author: Updated Author
Genre: Fiction


[Process completed - press Enter]
```

Lab Tasks= 5

Student Class Implementation

1.Define a class called Student with the following features:

• Private member variables: name (string), rollNumber (int), and

Grade (char).

• Default constructor: Initializes name to "Unknown", rollNumber

To 0, and grade to F.

• Parameterized constructor: Allows initializing name, rollNumber,

And grade with provided values.

• Method: void display() – Displays the name, roll number, and

Grade of the student.

2.Implement the Student class according to the above specifications.

3.In the main function:

- **Create two Student objects, one using the default constructor and**

Another using the parameterized constructor with name “Niamat”,

Roll number 101, and grade ‘A’.

Display the details of both students using the display() method.

4.Compile and run your program to verify its correctness.

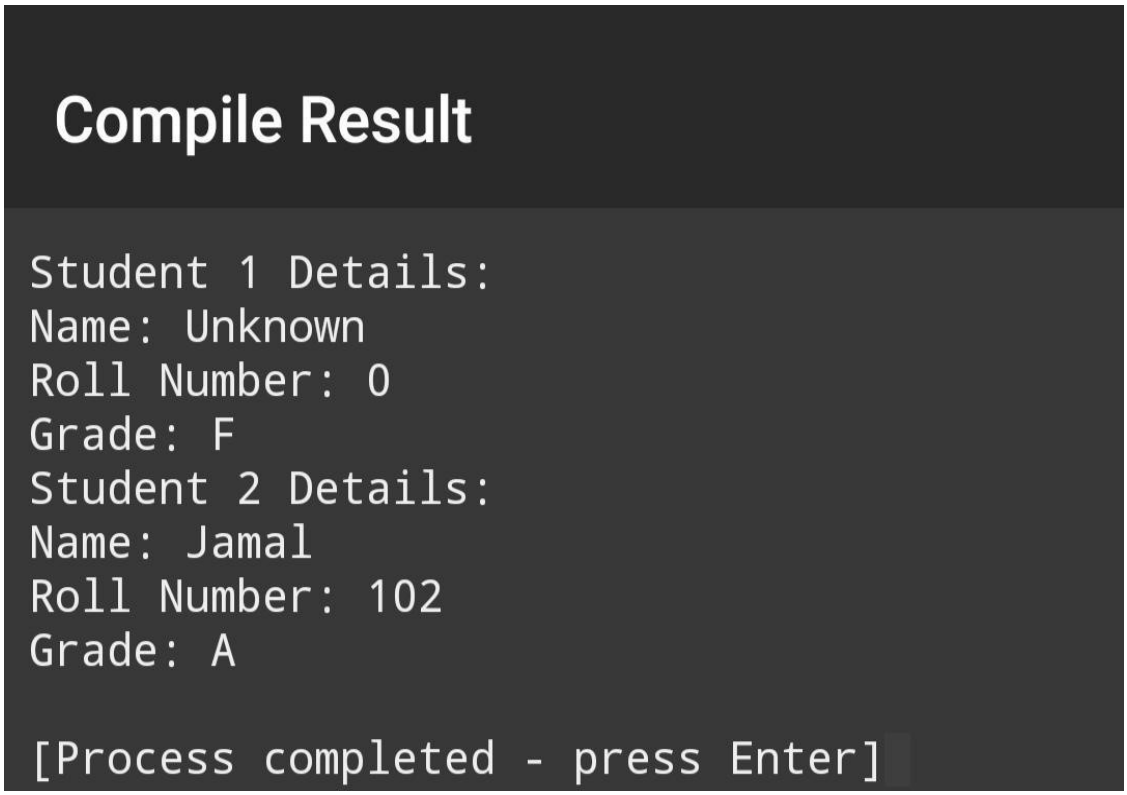
Program..

```
#include <iostream>
#include <string>
Using namespace std;
Class Student { Private:
    String name;
    Int rollNumber;
    Char grade;

Public:
```

```
   Student() {
        Name = “Unknown”;
rollNumber = 0;        grade
= ‘F’;
   }
   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(“Jamal”, 102, ‘A’);
   Cout << “Student 1 Details:” << endl;
   Student1.display();
```

```
    Cout << “Student 2 Details:” << endl;
    Student2.display();

    Return 0;
}
```

Out put..



Lab Tasks= 6

Implementing a Product Class with Overloaded Constructors

Problem Description:

You are required to create a class called Product to represent products in

A store. The Product class should have the following features:

1.Private Member Variables:

- **name (string): Name of the product.**
- **id (string): Unique identifier of the product.**
- **price (double): Price of the product.**
- **quantity (int): Quantity of the product in stock.**
- **category (string): Category of the product.**

2.Default Constructor:

- **Initializes name to "Unknown", id to "0000", price to 0.0, quantity**

To 0, and category to "Unknown".

3.Parameterized Constructors:

- **Constructor 1: Initializes name, id, price, quantity, and category**

With provided values.

- **Constructor 2: Initializes name, id, price, and category with**

Provided values, and sets quantity to 0.

- **Constructor 3: Initializes name, id, price, and quantity with**

Provided values, and sets category to "Unknown".

- **Constructor 4: Initializes name, price, quantity, and category with**

Provided values, and sets id to "0000".

- **Constructor 5: Initializes id, price, quantity, and category with Provided values, and sets name to "Unknown".**

4.Method: void display():

- **Displays the details of the product, including name, id, price,**

Quantity, and category.

Your Task:

1.Implement the Product class according to the above specifications.

2.In the main function:

- **Create five Product objects using each of the overloaded**

Constructors.

- **Display the details of each product using the display() method.**

3.Compile and run your program to verify its correctness.

Program..

```
#include <iostream>
#include <string>
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(const string &name, const string &id, double price, int quantity, const string &category)

: name(name), id(id), price(price), quantity(quantity), category(category) {}

// Parameterized Constructor 2

Product(const string &name, const string &id, double price, const string &category)

: name(name), id(id), price(price), quantity(0), category(category) {}

Product(const string &name, const string &id, double price, int quantity)

: name(name), id(id), price(price), quantity(quantity), category(“Unknown”) {}

```
Product(const string &name, double price, int quantity,const string &category)
                : name(name), id("0000"), price(price), quantity(quantity), category(category) {}

Product(const string &id, double price, int quantity, string &category)
                : name("Unknown"), id(id), price(price), quantity(quantity), category(category) {}

Void display() const {
    Cout << "Name: " << name << endl;
    Cout << "ID: " << id << endl;
    Cout << "Price: $" << price << endl;
    Cout << "Quantity: " << quantity << endl;
    Cout << "Category: " << category << endl;
}
};

Int main() {

Product product1;
Product product2("Laptop", "1234", 1500.00, 10, "Electronics");
```

Product product3(“Phone”, “5678”, 700.00, “Electronics”);

Product product4(“Tablet”, “9101”, 300.00, 5);

Product product5(“Accessory”, 50.00, 20, “Electronics”);

Product product6(“4321”, 20.00, 50, “Accessories”);

Cout << “Product 1:” << endl;

Product1.display();

Cout << endl;

Cout << “Product 2:” << endl;

Product2.display();
Cout << endl;

Cout << “Product 3:” << endl;

Product3.display();

Cout <<endl;

Cout << “Product 4:” << endl;

Product4.display();

Cout << endl;

Cout << “Product 5:” << endl;

Product5.display();

Cout << endl;

```
    Cout << “Product 6:” <<endl;
    Product6.display();
    Cout << endl;

    Return 0;
}
```

Out put..

Compile Result

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

Product 2:
Name: Laptop
ID: 1234
Price: $1500
Quantity: 10
Category: Electronics

Product 3:
Name: Phone
ID: 5678
Price: $700
Quantity: 0
Category: Electronics

Product 4:
Name: Tablet
ID: 9101
Price: $300
Quantity: 5
Category: Unknown

Product 5:
Name: Accessory
ID: 0000
Price: $50
Quantity: 20
Category: Electronics

Product 6:
Name: 4321
ID: 0000
Price: $20
Quantity: 50
Category: Accessories
```

Lab Tasks= 7

Implementing a Simple Counter Class with

Constructors and Destructor

Problem Description:

You are tasked with creating a simple class called Counter to represent a

Counter. The Counter class should have the following features:

1.Private Member Variable:

- **count (integer): Represents the count value of the counter.**

2.Default Constructor:

- **Initializes count to 0.**

- **Displays a message indicating that the counter has been initialized**

With 0.

3.Parameterized Constructor:

- **Allows initializing count with a provided initial value.**
- **Displays a message indicating the initial value with which the**

Counter has been initialized.

4.Destructor:

- **Displays a message indicating that the destructor has been called**

Along with the final count value when the counter object is

Destroyed.

5.Method: void increment():

- **Increments the count value by 1.**

6.Method: void display():

- **Displays the current count value.**

Your Task:

Implement the Counter class according to the provided specifications.

In the main function:

Create two Counter objects, one using the default constructor and

Another using the parameterized constructor with an initial count

Value of 5.

- **Increment the count value of both counters using the increment()**

Method.

- **Display the count value of both counters using the display() method.**

Compile and run your program to verify its correctness.

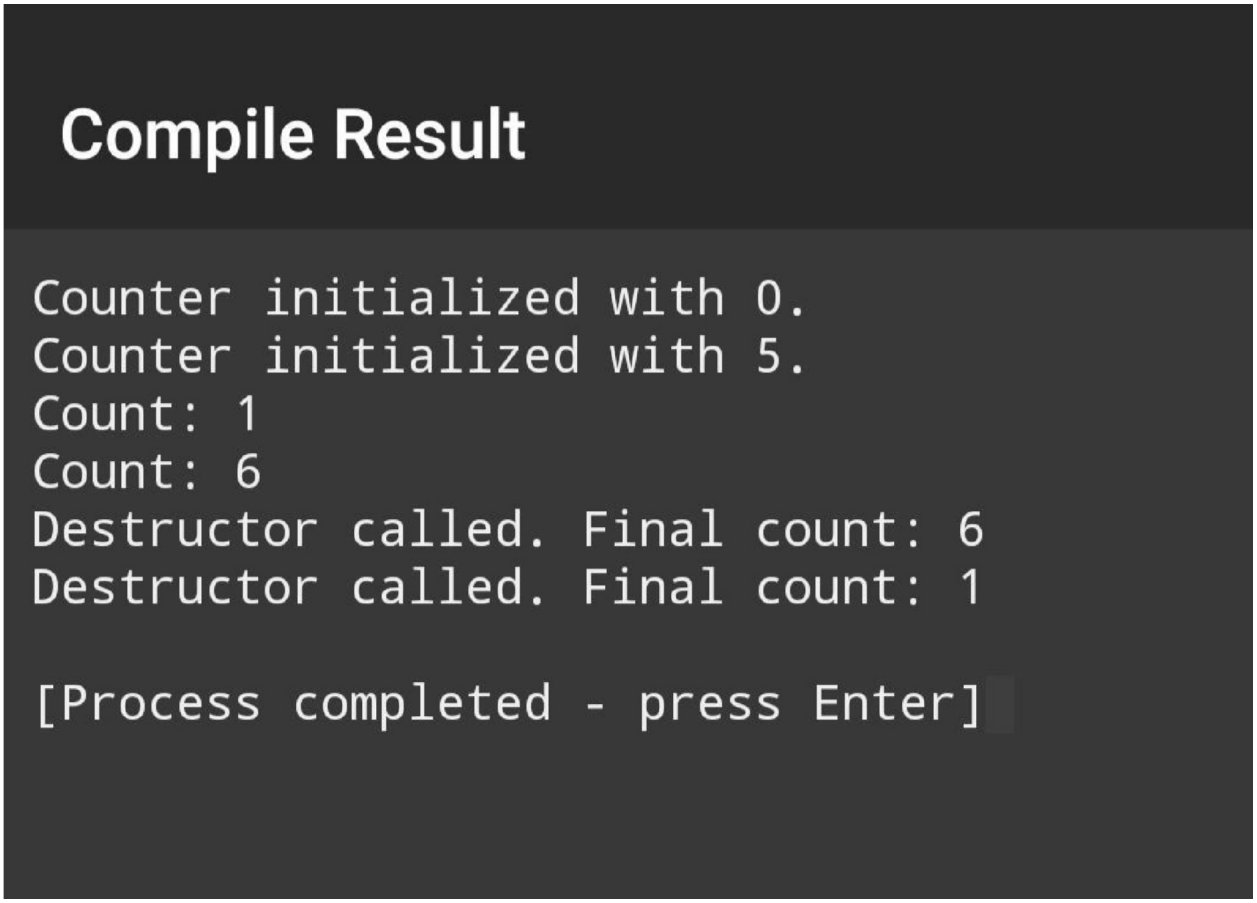
Program..

```
#include <iostream>
Using namespace std;
Class Counter { Private:
```

```
    Int count;

Public:

    Counter() {
        Count = 0;
        Cout << “Counter initialized with 0.” << endl;
    }
    Counter(int initialCount) {
        Count = initialCount;
        Cout << “Counter initialized with “ << initialCount << “.” << endl;
    }
    ~Counter() {
       Cout << “Destructor called. Final count: “ << count << endl;
    }
    Void increment() {
        Count++;
    }
    Void display() {
        Cout << “Count: “ << count <<endl;
    }
};

Int main() {
```

```
    Counter counter1;
    Counter counter2(5);
    Counter1.increment();
    Counter2.increment();
    Counter1.display();
    Counter2.display();

    Return 0;
}
```

Out put..



Lab Tasks= 8

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.

Detailed Requirements:

1.Define a class Book with the following specifications:

- **Private data members:**

- **static int nextISBN: Static data member to store the next**

Available ISBN.

- **int ISBN: Unique ISBN of the book.**

- **string title: Title of the book.**

• string author: Author of the book.

• Public member functions:

• A constructor to initialize the title, author, and ISBN of the

Book using the nextISBN, and then increment nextISBN for

The next book.

• Member functions to set and get the title and author of the

Book.

• A member function displayDetails() to display the details of

The book, including its title, author, and ISBN.

2.Create three objects of class Book, each representing a different book, And register them in the library catalog.

3.Display the details of each book, including its title, author, and ISBN.

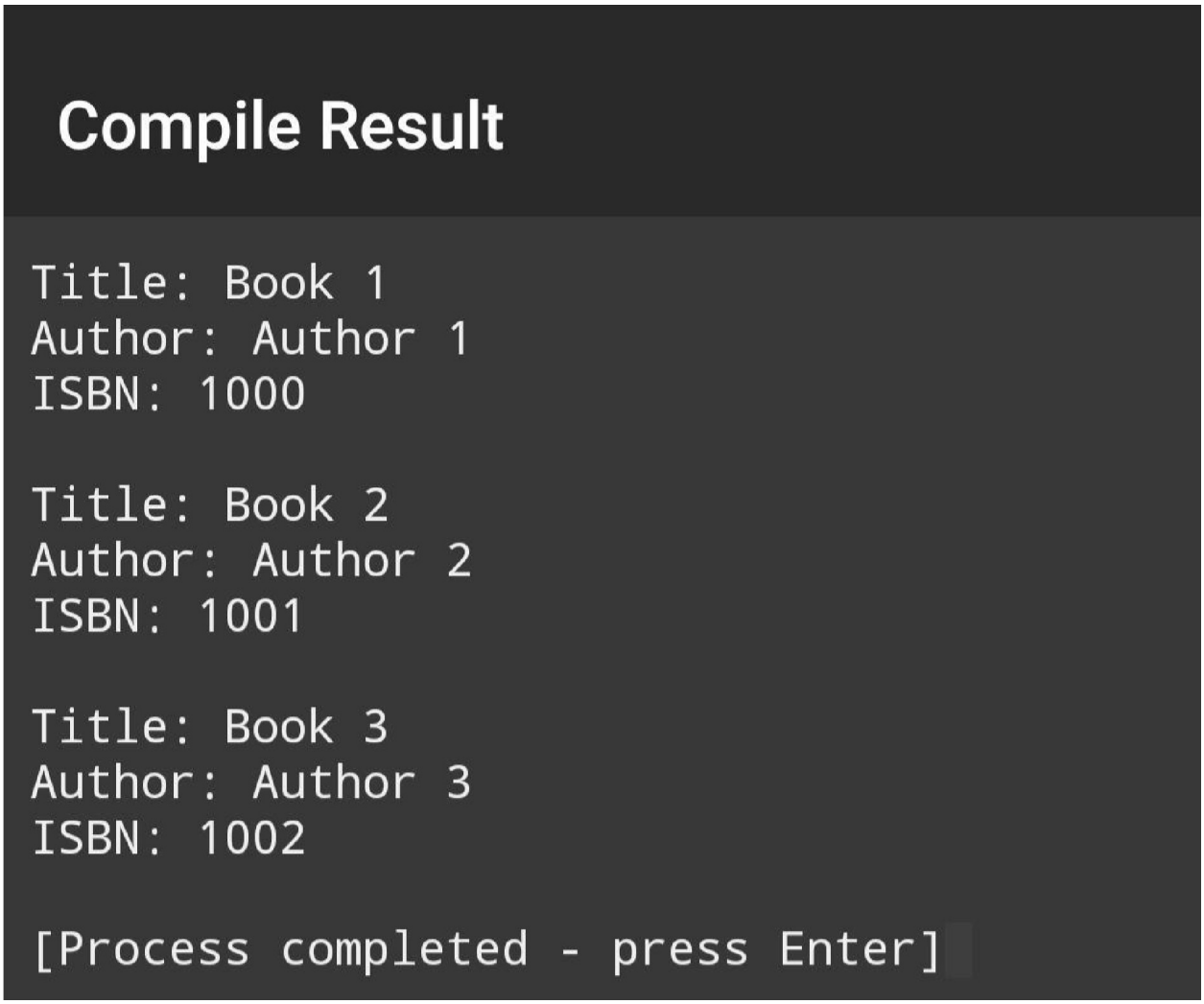
Program..

```
#include <iostream>
#include <string>
Using namespace std;
Class Book { Private:
   Static int nextISBN;
   Int ISBN;
   String title;
   String author;

Public:
                                Book(string t,string a) {
      Title = t;
            Author  =  a;
ISBN     =     nextISBN;
nextISBN++;
   }

   Void setTitle(string t) {
```

```
    Title = t;
}


String getTitle() {
    Return title;
}


Void setAuthor(string a) {
    Author = a;
}


String getAuthor() {
    Return author;
}


Void displayDetails() {
    Cout << “Title: “ << title << endl;
    Cout << “Author: “ << author << endl;
    Cout << “ISBN: “ << ISBN << endl;
}
```

```
};

Int Book::nextISBN = 1000;

Int main() {
   Book book1("Book 1", "Author 1");
   Book book2("Book 2", "Author 2");
   Book book3("Book 3", "Author 3");


   Book1.displayDetails();
   Cout << std::endl;
   Book2.displayDetails();
   Cout << std::endl;
   Book3.displayDetails();

   Return 0;
}
```

Out put..



LAB TASKS= 9

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.

Detailed Requirements:

1.Define a class StringUtils with the following static member function:

- **int countVowels(const string& str): Calculate and return the**

Count of vowels (a, e, I, o, u) in the given string str.

2.Implement the static member function countVowels of the

StringUtils class.

3.In the main function of the program:

- **Prompt the user to enter a string.**

- **Use the countVowels function to calculate the count of vowels in**

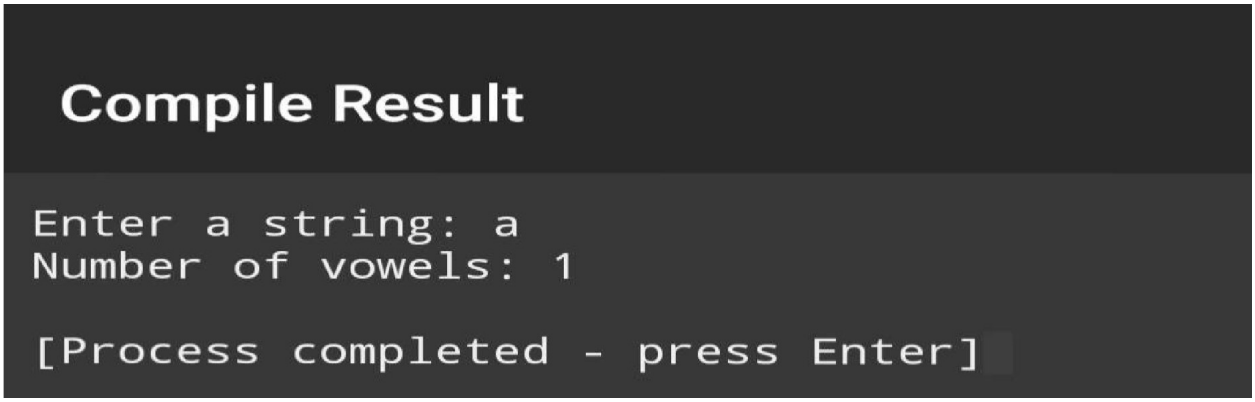
The input string and display the result.

Program..

```
#include <iostream>
#include <string>
Using namespace std ;
Class StringUtils { Public:
   Static int countVowels(const string& str) {
      Int count = 0;
      For (char c : str) {
         C = tolower©;
         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;
}
```

Out put..



Lab Tasks= 10

Task Description:

Create a program to manage two numbers and perform basic arithmetic

Operations on them. There are two classes involved: Number and

Calculator.

Number class should have:

- **Private data members for two numbers: num1 and num2.**

- **Public member functions to set and get the numbers: setNumber1, setNumber2, getNumber1, getNumber2.**

- **Friend function to allow Calculator to perform arithmetic operations**

On these numbers: friend calculate(Number&).

Calculator class should have:

- **No data members.**
- **Public member functions to perform addition, subtraction,**

Multiplication, and division of two numbers: add, subtract, multiply,

Divide.

Your program should allow the user to input two numbers, perform

Arithmetic operations on them, and display the results.

Note: Ensure that Calculator can access the numbers of Number objects

Using the friend function mechanism.

Program..