NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

1. Calculate the sum of all natural numbers from 1 to n, where n is a positive integer. Use the formula:
   Sum=n×(n+1)/2 .
   Take n as input and output the sum of natural numbers from 1 to n .

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

int main() {
    int n;
    cin >> n;

    int sum = n * (n + 1) / 2;

    cout << sum << endl;

    return 0;
```
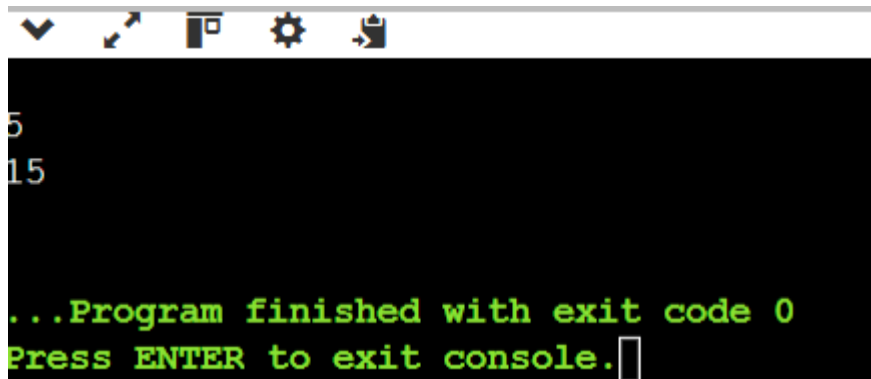
NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

2. Check if a given number n is a prime number. A prime number is a natural number greater than 1 that has no positive divisors other than 1 and itself.

To determine if a number is prime, iterate from 2 to √n and check n is divisible by any number in this range. If it is divisible, it is not a prime number; otherwise, it is a prime.

CODE-

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

bool isPrime(int n) {
    if (n <= 1) return false;

    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) {
            return false;
        }
    }
```
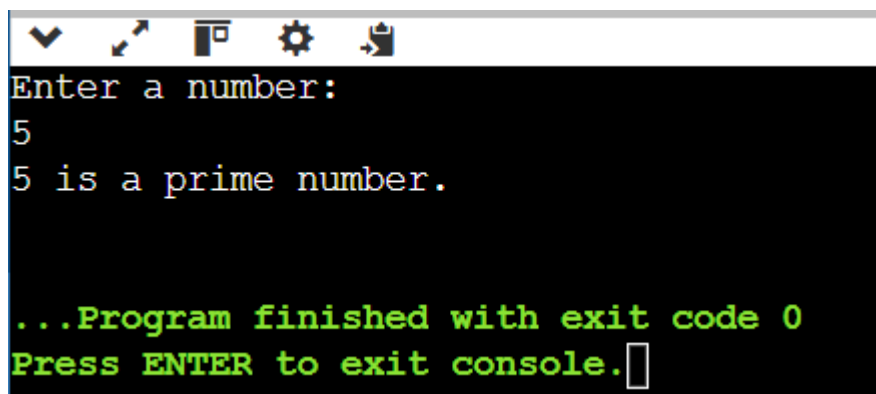
NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
  }
  return true;
}


int main() {
  int n;
  cout << "Enter a number: ";
  cin >> n;


  if (isPrime(n)) {
    cout << n << " is a prime number." << endl;
  } else {
    cout << n << " is not a prime number." << endl;
  }


  return 0;
}
```



```
Enter a number:
5
5 is a prime number.



...Program finished with exit code 0
Press ENTER to exit console.
```

3. Print all odd numbers between 1 and n, inclusive. Odd numbers are integers that are not divisible by 2. These numbers should be printed in ascending order, separated by spaces.

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

This problem is a simple introduction to loops and conditional checks. The goal is to use a loop to iterate over the numbers and check if they are odd using the condition i%2≠0.

CODE-

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

int main() {
    int n;
    cin >> n; // Input the value of n

    for (int i = 1; i <= n; i++) { // Loop from 1 to n
        if (i % 2 != 0) { // Check if the number is odd
            cout << i << " "; // Print the odd number
        }
    }

    return 0;
}
```

```
5
1 3 5

...Program finished with exit code 0
Press ENTER to exit console.
```

4. Calculate the sum of all odd numbers from 1 to n. An odd number is an integer that is not divisible by 2.The sum of odd numbers, iterate through all the numbers from 1 to n, check if each number is odd, and accumulate the sum.

CODE -

```cpp
#include <iostream>

using namespace std;

int main() {
    int n, sum = 0;
    cin >> n;

    for (int i = 1; i <= n; i++) { // Loop from 1 to n
        if (i % 2 != 0) { // Check if the number is odd
            sum += i; // Add the odd number to the sum
        }
```

```
    }


    cout << sum << endl; // Output the total sum

    return 0;

}
```



```
5
9

...Program finished with exit code 0
Press ENTER to exit console.
```

5. Print the multiplication table of a given number n. A multiplication table
for a number n is a list of products of n with integers from 1 to 10. For
example, the multiplication table for 3 is:
3×1=3,3×2=6,…,3×10=30.

CODE –

```cpp
#include <iostream>

using namespace std;

int main() {
    int n;

    cin >> n; // Input the number for which the multiplication table is needed
```
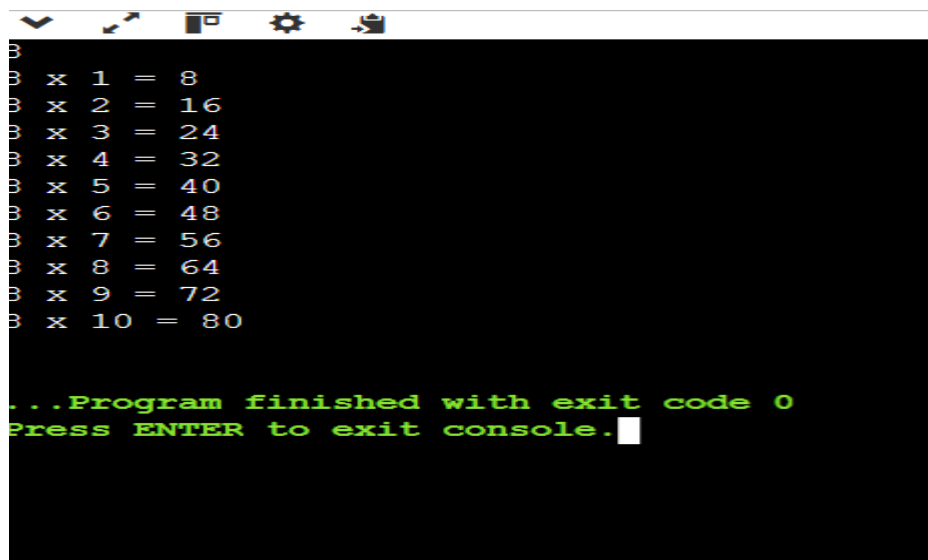
```
    for (int i = 1; i <= 10; i++) { // Loop from 1 to 10

        cout << n << " x " << i << " = " << n * i << endl; // Print the product

    }


    return 0;

}
```

```
B
B x 1 = 8
B x 2 = 16
B x 3 = 24
B x 4 = 32
B x 5 = 40
B x 6 = 48
B x 7 = 56
B x 8 = 64
B x 9 = 72
B x 10 = 80


...Program finished with exit code 0
Press ENTER to exit console.
```

6. 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 to determine how many digits
   are present in n. This task will help you practice working with
   loops, number manipulation, and conditional logic.

CODE –

#include <iostream>

NAME –Ankur Chauhan

UID – 22BCS14662

SECTION – FL-IOT-603-B

```cpp
using namespace std;

int main() {

    int n, count = 0;

    cin >> n; // Input the number


    if (n == 0) {

        count = 1; // Special case: 0 has 1 digit

    } else {

        while (n != 0) {

            n = n / 10; // Remove the last digit

            count++;    // Increment the count

        }

    } cout << count << endl; // Output the total number of digits

    return 0;
```
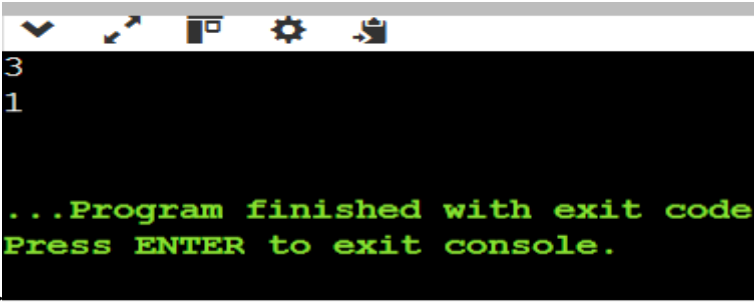


```
3
1

...Program finished with exit code
Press ENTER to exit console.
```

```
}
```

7. Reverse the digits of a given number n. For example, if the input number is 12345, the output should be 54321. The task involves using loops and modulus operators to extract the digits and construct the reversed number.

CODE –

#include <iostream>
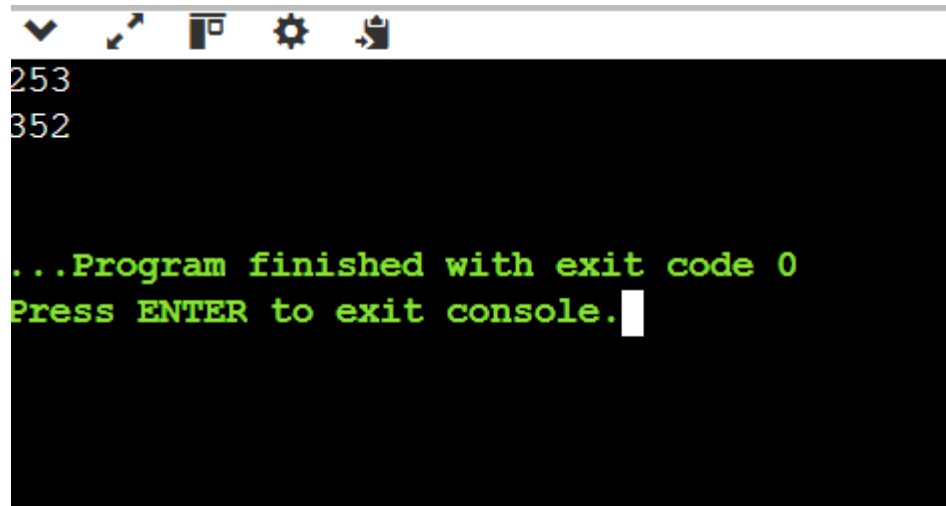
NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
using namespace std;

int main() {
    int n, reversed = 0;
    cin >> n; // Input the number

    while (n != 0) {
        int digit = n % 10;        // Extract the last digit
        reversed = reversed * 10 + digit; // Append the digit to the reversed number
        n = n / 10;              // Remove the last digit from the number
    }

    cout << reversed << endl; // Output the reversed number
    return 0;
}
```

```
253
352


...Program finished with exit code 0
Press ENTER to exit console.
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

8. Find the largest digit in a given number n. For example, for the number 2734, the largest digit is 7. You need to extract each digit from the number and determine the largest one. The task will involve using loops and modulus operations to isolate the digits.

CODE –

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

int main() {
    int n, largest = 0;
    cin >> n; // Input the number

    while (n > 0) {
        int digit = n % 10; // Extract the last digit
        if (digit > largest) {
            largest = digit; // Update largest if the current digit is greater
        }
        n = n / 10; // Remove the last digit from the number
    }

    cout << largest << endl; // Output the largest digit
    return 0;
}
```
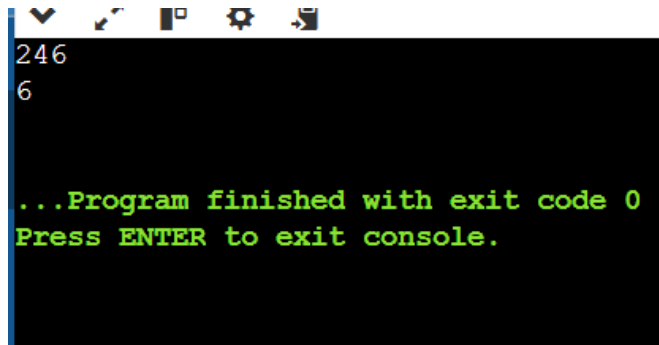
9. Check whether a given number is a palindrome or not. A number is called a palindrome if it reads the same backward as forward. For example, 121 is a palindrome because reading it from left to right is the same as reading it from right to left. Similarly, 12321 is also a palindrome, but 12345 is not.

CODE –

```cpp
#include <iostream>

using namespace std;

int main() {
    int n, reversed = 0, original, digit;

    cin >> n;  // Input the number

original = n;  // Store the original number

while (n > 0) {

    digit = n % 10;          // Extract the last digit

    reversed = reversed * 10 + digit;  // Build the reversed number

    n = n / 10;              // Remove the last digit from the number

  }

 if (original == reversed) {

    cout << "Palindrome" << endl;  // If the original number is equal to the reversed one, it is a palindrome

  } else {
```
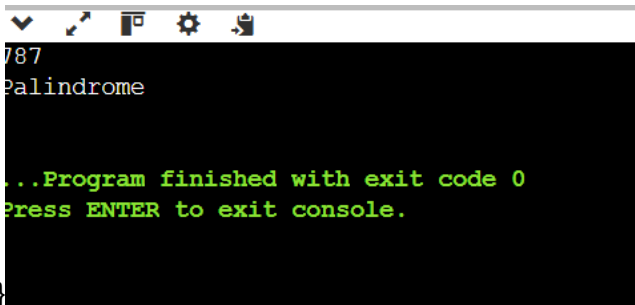
```
    cout << "Not Palindrome" << endl;  // If the numbers are not equal, it's not a
palindrome

  }
return 0;
```


10. Calculate the sum of the digits of a given number n. For example, for the number 12345, the sum of the digits is 1+2+3+4+5=15. To solve this, you will need to extract each digit from the number and calculate the total sum.

CODE -

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

int main() {
    int n, sum = 0, digit;
    cin >> n;  // Input the number

    while (n > 0) {
        digit = n % 10;   // Extract the last digit
        sum += digit;     // Add the digit to the sum
        n = n / 10;       // Remove the last digit from the number
    }
```
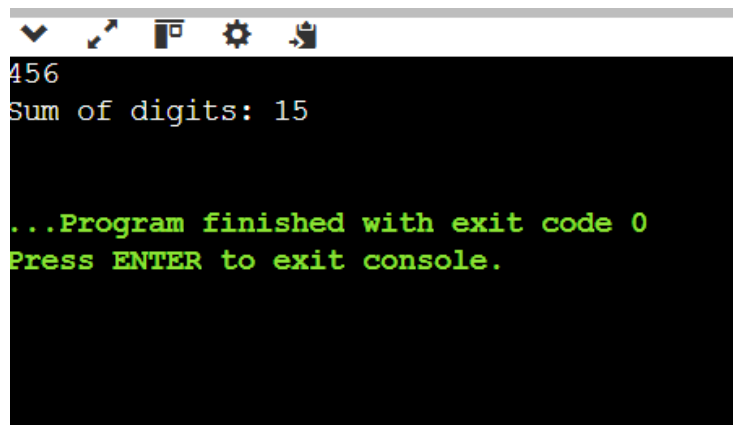
```
cout << "Sum of digits: " << sum << endl;  // Output the sum of digits


    return 0;

}
```



```
456
Sum of digits: 15


...Program finished with exit code 0
Press ENTER to exit console.
```

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

CODE –

```
#include <iostream>

using namespace std;


// Function to calculate the area of a circle

float area(float radius) {

    return 3.14159 * radius * radius;  // Area of circle = π * radius^2

}


// Function to calculate the area of a rectangle

float area(float length, float width) {
```

```cpp
    return length * width;  // Area of rectangle = length * width

 }


// Function to calculate the area of a triangle

float area(float base, float height) {

    return 0.5 * base * height;  // Area of triangle = 0.5 * base * height

 }


int main() {

    float radius, length, width, base, height;


    // Calculate area of circle

    cout << "Enter radius of circle: ";

    cin >> radius;

    cout << "Area of the circle: " << area(radius) << endl;


    // Calculate area of rectangle

    cout << "Enter length and width of rectangle: ";

    cin >> length >> width;

    cout << "Area of the rectangle: " << area(length, width) << endl;


    // Calculate area of triangle

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

    cin >> base >> height;

    cout << "Area of the triangle: " << area(base, height) << endl;


    return 0;
```
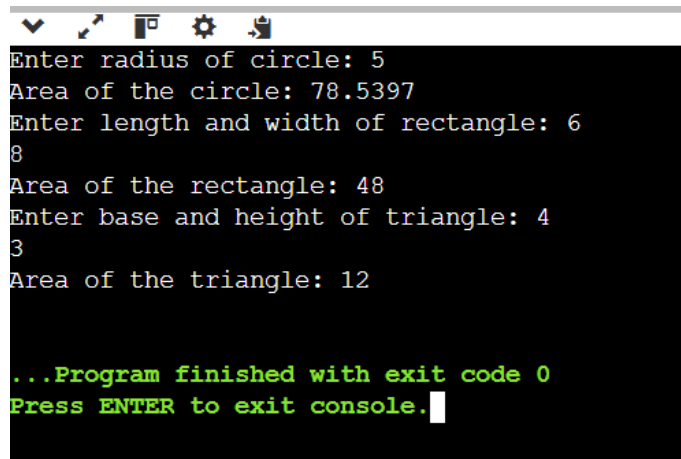
# NAME –Ankur Chauhan
# UID – 22BCS14662
# SECTION – FL-IOT-603-B
}

12. Write a program that demonstrates function overloading to calculate the salary of employees at different levels in a company hierarchy. Implement overloaded functions to compute salary for:

Intern (basic stipend).

Regular employee (base salary + bonuses).

Manager (base salary + bonuses + performance incentives).

CODE –

```
#include <iostream>

using namespace std;


// Function to calculate salary for an intern (basic stipend)

float calculateSalary(float stipend) {

    return stipend;  // Intern gets a basic stipend

}


// Function to calculate salary for a regular employee (base salary + bonuses)
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
float calculateSalary(float baseSalary, float bonuses) {

    return baseSalary + bonuses;  // Regular employee gets base salary + bonuses

}


// Function to calculate salary for a manager (base salary + bonuses + performance incentives)

float calculateSalary(float baseSalary, float bonuses, float performanceIncentives) {

    return baseSalary + bonuses + performanceIncentives;  // Manager gets base salary + bonuses + performance incentives

}


int main() {

    float stipend, baseSalary, bonuses, performanceIncentives;


    // Calculate salary for Intern

    cout << "Enter stipend for Intern: ";

    cin >> stipend;

    cout << "Intern's salary: " << calculateSalary(stipend) << endl;


    // Calculate salary for Regular Employee

    cout << "Enter base salary for Regular Employee: ";

    cin >> baseSalary;

    cout << "Enter bonuses for Regular Employee: ";

    cin >> bonuses;

    cout << "Regular Employee's salary: " << calculateSalary(baseSalary, bonuses) << endl;


    // Calculate salary for Manager
```

```cpp
    cout << "Enter base salary for Manager: ";

    cin >> baseSalary;

    cout << "Enter bonuses for Manager: ";

    cin >> bonuses;

    cout << "Enter performance incentives for Manager: ";

    cin >> performanceIncentives;

    cout << "Manager's salary: " << calculateSalary(baseSalary, bonuses,
performanceIncentives) << endl;


    return 0;

}
```
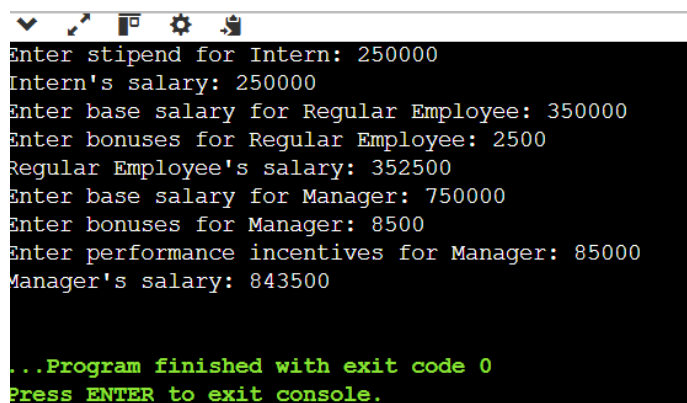

13. Create a program that demonstrates inheritance by defining:

A base class Student to store details like Roll Number and Name.

A derived class Result to store marks for three subjects and calculate the total and percentage.

CODE –

```cpp
#include <iostream>

#include <string>

using namespace std;
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
// Base class: Student

class Student {

protected:

    int rollNumber;

    string name;


public:

    // Constructor to initialize roll number and name

    Student(int r, string n) {

        rollNumber = r;

        name = n;

    }


    // Function to display student details

    void displayDetails() {

        cout << "Roll Number: " << rollNumber << endl;

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

    }
};


// Derived class: Result (inherits from Student)

class Result : public Student {

private:

    float marks1, marks2, marks3; // Marks for three subjects

    float total, percentage;


public:
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
    // Constructor to initialize marks for the subjects, and base class constructor

    Result(int r, string n, float m1, float m2, float m3) : Student(r, n) {

        marks1 = m1;

        marks2 = m2;

        marks3 = m3;

        total = 0;

        percentage = 0.0;

    }


    // Function to calculate total and percentage

    void calculateResult() {

        total = marks1 + marks2 + marks3;

        percentage = (total / 300) * 100; // Assuming 300 is the maximum marks

    }


    // Function to display result (total and percentage)

    void displayResult() {

        calculateResult();

        cout << "Total Marks: " << total << "/300" << endl;

        cout << "Percentage: " << percentage << "%" << endl;

    }
};


int main() {

    int rollNumber;

    string name;

    float marks1, marks2, marks3;
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
    // Input student details
    cout << "Enter Roll Number: ";

    cin >> rollNumber;

    cin.ignore(); // To clear the input buffer

    cout << "Enter Name: ";

    getline(cin, name);

    cout << "Enter Marks for Subject 1: ";

    cin >> marks1;

    cout << "Enter Marks for Subject 2: ";

    cin >> marks2;

    cout << "Enter Marks for Subject 3: ";

    cin >> marks3;


    // Create an object of Result (which also calls Student class constructor)

    Result studentResult(rollNumber, name, marks1, marks2, marks3);


    // Display student details and result

    cout << endl;

    studentResult.displayDetails();

    studentResult.displayResult();


    return 0;

}
```
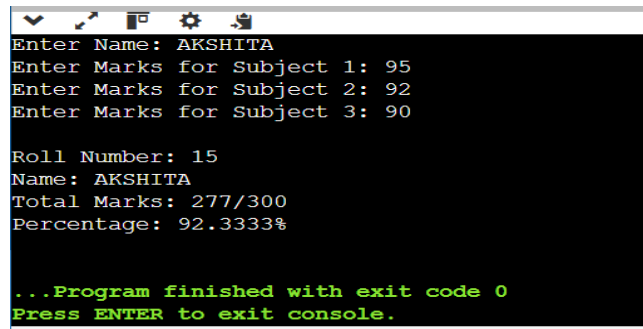
NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B


```
Enter Name: AKSHITA
Enter Marks for Subject 1: 95
Enter Marks for Subject 2: 92
Enter Marks for Subject 3: 90

Roll Number: 15
Name: AKSHITA
Total Marks: 277/300
Percentage: 92.3333%


...Program finished with exit code 0
Press ENTER to exit console.
```

14. Create a program that demonstrates polymorphism by calculating the area
    of different shapes using a base class Shape and derived classes for Circle,
    Rectangle, and Triangle. Each derived class should override a virtual
    function to compute the area of the respective shape.

CODE –

```cpp
#include <iostream>

#include <cmath>

using namespace std;


// Base class: Shape

class Shape {

public:

    // Virtual function to calculate area, will be overridden by derived classes

    virtual double area() const = 0;


    // Virtual destructor

    virtual ~Shape() {}

};
```

```cpp
// Derived class: Circle

class Circle : public Shape {

private:

    double radius;


public:

    Circle(double r) : radius(r) {}


    // Override the area function for Circle

    double area() const override {

        return 3.14159 * radius * radius; // πr^2

    }

};


// Derived class: Rectangle

class Rectangle : public Shape {

private:

    double length, width;


public:

    Rectangle(double l, double w) : length(l), width(w) {}


    // Override the area function for Rectangle

    double area() const override {

        return length * width; // length * width

    }

};
```

```cpp
// Derived class: Triangle

class Triangle : public Shape {

private:

    double base, height;


public:

    Triangle(double b, double h) : base(b), height(h) {}


    // Override the area function for Triangle

    double area() const override {

        return 0.5 * base * height; // (base * height) / 2

    }

};


int main() {
    // Create objects for each shape

    Shape* shapes[3];

    shapes[0] = new Circle(5); // Circle with radius 5

    shapes[1] = new Rectangle(4, 6); // Rectangle with length 4 and width 6

    shapes[2] = new Triangle(4, 3); // Triangle with base 4 and height 3


    // Output the areas of the shapes

    for (int i = 0; i < 3; ++i) {

        cout << "Area of shape " << i + 1 << ": " << shapes[i]->area() << endl;

    }
```
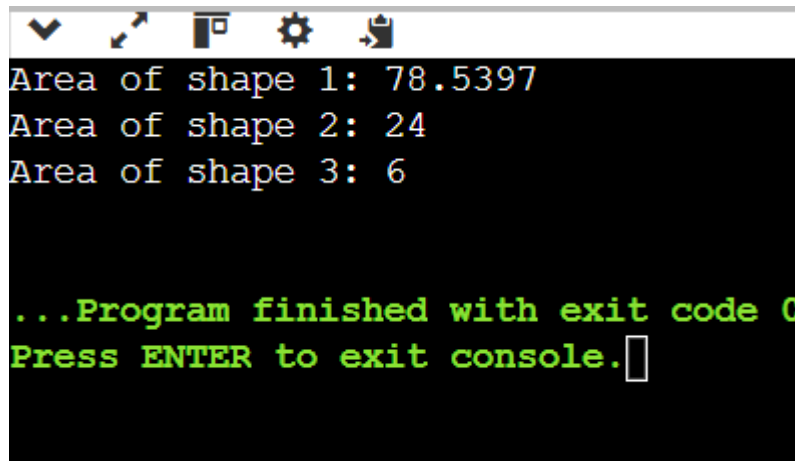
```
    // Clean up dynamic memory

    for (int i = 0; i < 3; ++i) {

        delete shapes[i];

    }


    return 0;

}
```



```
Area of shape 1: 78.5397
Area of shape 2: 24
Area of shape 3: 6


...Program finished with exit code 0
Press ENTER to exit console.
```

15. 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.

CODE-

```cpp
#include <iostream>

#include <cmath>

using namespace std;


// Base class: Shape

class Shape {
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
public:
    // Virtual function to calculate area
    virtual double area() const = 0;


    // Virtual destructor
    virtual ~Shape() {}
};


// Derived class: Circle
class Circle : public Shape {
private:
    double radius;


public:
    Circle(double r) : radius(r) {}

    // Override the area function for Circle
    double area() const override {
        return 3.14159 * radius * radius; // πr^2
    }
};


// Derived class: Rectangle
class Rectangle : public Shape {
private:
    double length, width;
```

```cpp
public:

    Rectangle(double l, double w) : length(l), width(w) {}


    // Override the area function for Rectangle

    double area() const override {

        return length * width; // length * width

    }

};


// Derived class: Triangle

class Triangle : public Shape {

private:

    double base, height;


public:

    Triangle(double b, double h) : base(b), height(h) {}


    // Override the area function for Triangle

    double area() const override {

        return 0.5 * base * height; // (base * height) / 2

    }

};


int main() {

    int choice;


    // Get user input to choose a shape
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
cout << "Choose a shape to calculate the area:" << endl;

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

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

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

cout << "Enter your choice (1/2/3): ";

cin >> choice;


Shape* shape = nullptr;


// Based on user's choice, create the appropriate shape
if (choice == 1) {

    double radius;

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

    cin >> radius;

    shape = new Circle(radius);  // Create a Circle object
}
else if (choice == 2) {

    double length, width;

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

    cin >> length >> width;

    shape = new Rectangle(length, width);  // Create a Rectangle object
}
else if (choice == 3) {

    double base, height;

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

    cin >> base >> height;

    shape = new Triangle(base, height);  // Create a Triangle object
```
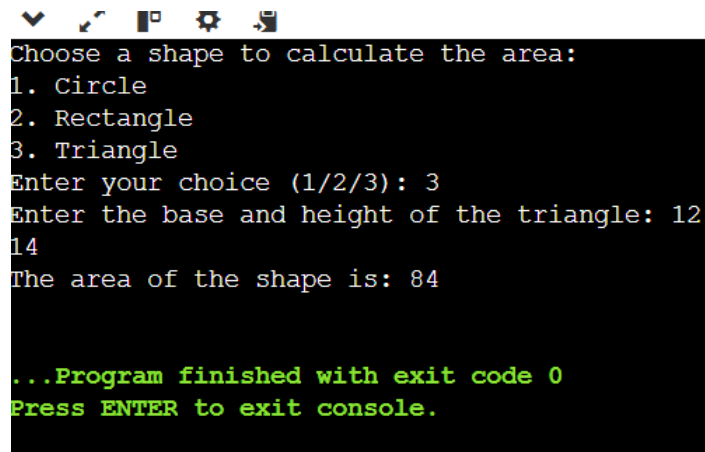
```cpp
    }

    else {

        cout << "Invalid choice!" << endl;

        return 1;

    }



    // Output the area of the selected shape

    cout << "The area of the shape is: " << shape->area() << endl;



    // Clean up dynamically allocated memory

    delete shape;



    return 0;

}
```

```
Choose a shape to calculate the area:
1. Circle
2. Rectangle
3. Triangle
Enter your choice (1/2/3): 3
Enter the base and height of the triangle: 12
14
The area of the shape is: 84


...Program finished with exit code 0
Press ENTER to exit console.
```

16. 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.

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

CODE –

```cpp
#include <iostream>

using namespace std;


// Function to perform matrix addition
void operate(int mat1[10][10], int mat2[10][10], int result[10][10], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            result[i][j] = mat1[i][j] + mat2[i][j];
        }
    }
}


// Function to perform matrix multiplication
void operate(int mat1[10][10], int mat2[10][10], int result[10][10], int rows1, int cols1, int rows2, int cols2) {
    // Check if multiplication is possible
    if (cols1 != rows2) {
        cout << "Matrix multiplication is not possible!" << endl;
        return;
    }

    // Perform matrix multiplication
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols2; j++) {
            result[i][j] = 0;
            for (int k = 0; k < cols1; k++) {
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
                result[i][j] += mat1[i][k] * mat2[k][j];
        }
      }
   }
}


// Function to display a matrix
void displayMatrix(int mat[10][10], int rows, int cols) {
   for (int i = 0; i < rows; i++) {
      for (int j = 0; j < cols; j++) {
         cout << mat[i][j] << " ";
      }
      cout << endl;
   }
}


int main() {
   int choice;
   cout << "Choose Matrix Operation:\n";
   cout << "1. Matrix Addition\n";
   cout << "2. Matrix Multiplication\n";
   cout << "Enter your choice (1/2): ";
   cin >> choice;


   int mat1[10][10], mat2[10][10], result[10][10];
   int rows1, cols1, rows2, cols2;
```

NAME –Ankur Chauhan

UID – 22BCS14662

SECTION – FL-IOT-603-B

```cpp
switch (choice) {

  case 1:

    // Matrix Addition

    cout << "Enter the number of rows and columns for matrix 1: ";

    cin >> rows1 >> cols1;

    cout << "Enter the elements of matrix 1:\n";

    for (int i = 0; i < rows1; i++) {

      for (int j = 0; j < cols1; j++) {

        cin >> mat1[i][j];

      }

    }


    cout << "Enter the number of rows and columns for matrix 2: ";

    cin >> rows2 >> cols2;

    cout << "Enter the elements of matrix 2:\n";

    for (int i = 0; i < rows2; i++) {

      for (int j = 0; j < cols2; j++) {

        cin >> mat2[i][j];

      }

    }


    if (rows1 == rows2 && cols1 == cols2) {

      operate(mat1, mat2, result, rows1, cols1);  // Matrix addition

      cout << "Result of Matrix Addition:\n";

      displayMatrix(result, rows1, cols1);

    } else {

      cout << "Matrix addition is not possible due to mismatched dimensions.\n";
```

```cpp
        }
        break;

    case 2:
        // Matrix Multiplication
        cout << "Enter the number of rows and columns for matrix 1: ";
        cin >> rows1 >> cols1;
        cout << "Enter the elements of matrix 1:\n";
        for (int i = 0; i < rows1; i++) {
            for (int j = 0; j < cols1; j++) {
                cin >> mat1[i][j];
            }
        }

        cout << "Enter the number of rows and columns for matrix 2: ";
        cin >> rows2 >> cols2;
        cout << "Enter the elements of matrix 2:\n";
        for (int i = 0; i < rows2; i++) {
            for (int j = 0; j < cols2; j++) {
                cin >> mat2[i][j];
            }
        }

        operate(mat1, mat2, result, rows1, cols1, rows2, cols2);  // Matrix multiplication
        cout << "Result of Matrix Multiplication:\n";
        displayMatrix(result, rows1, cols2);
        break;
```
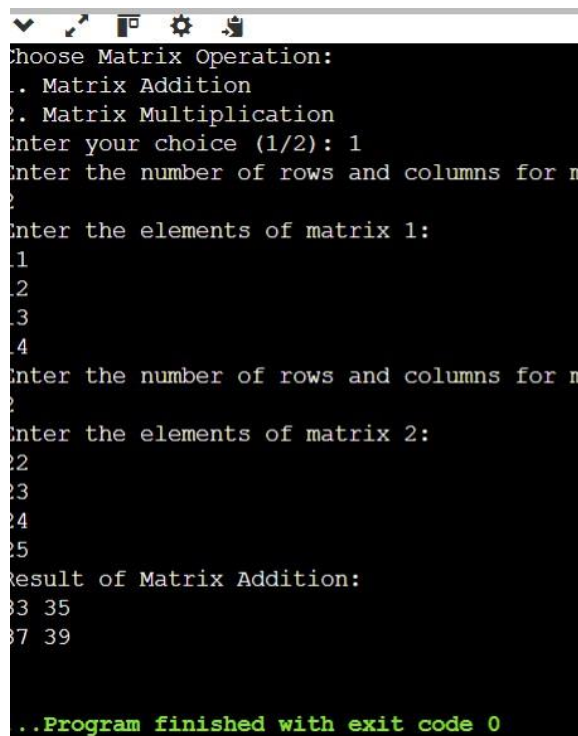
```
    default:

        cout << "Invalid choice.\n";

        break;

    }


    return 0;

}
```


17. 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).

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

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

CODE –

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

// Base class for storing book details
class Book {
protected:
    string title;
    string author;
    string ISBN;
public:
    // Constructor to initialize book details
    Book(string t, string a, string isbn) {
        title = t;
        author = a;
        ISBN = isbn;
    }

    // Function to display book details
    void displayBookDetails() {
        cout << "Book Title: " << title << endl;
        cout << "Author: " << author << endl;
        cout << "ISBN: " << ISBN << endl;
```

```cpp
    }
 };


// Base class for storing borrower details

class Borrower {

protected:

    string name;

    int id;

    string borrowedBook;

public:

    // Constructor to initialize borrower details

    Borrower(string n, int i) {

        name = n;

        id = i;

        borrowedBook = "";

    }


    // Function to borrow a book

    void borrowBook(string bookTitle) {

        borrowedBook = bookTitle;

    }


    // Function to return a book

    void returnBook() {

        borrowedBook = "";

    }
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
  // Function to display borrower details

  void displayBorrowerDetails() {

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

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

    if (borrowedBook != "") {

      cout << "Borrowed Book: " << borrowedBook << endl;

    } else {

      cout << "No book borrowed." << endl;

    }

  }

};


// Derived class that inherits from both Book and Borrower

class Library : public Book, public Borrower {

public:

  // Constructor to initialize both Book and Borrower details

  Library(string bookTitle, string bookAuthor, string bookISBN, string borrowerName, int borrowerID)

    : Book(bookTitle, bookAuthor, bookISBN), Borrower(borrowerName, borrowerID) {}


  // Function to simulate borrowing a book

  void borrow() {

    if (borrowedBook == "") {

      borrowBook(title);

      cout << name << " borrowed the book: " << title << endl;

    } else {

      cout << name << " already borrowed a book: " << borrowedBook << endl;
```

```cpp
        }
    }


    // Function to simulate returning a book
    void returnBack() {
        if (borrowedBook != "") {
            cout << name << " returned the book: " << borrowedBook << endl;
            returnBook();
        } else {
            cout << name << " has no book to return." << endl;
        }
    }


    // Function to display all details
    void displayDetails() {
        displayBookDetails();
        displayBorrowerDetails();
    }
};


int main() {
    // Creating a library object with book and borrower details
    Library lib("The Great Gatsby", "F. Scott Fitzgerald", "978-0743273565", "John Doe", 101);


    // Displaying initial details
    cout << "Library System Simulation:" << endl;
```

```cpp
    lib.displayDetails();


    cout << "\nAttempting to borrow the book..." << endl;

    lib.borrow(); // Borrow the book


    cout << "\nAttempting to borrow the book again..." << endl;

    lib.borrow(); // Try to borrow another book while already borrowing one


    cout << "\nAttempting to return the book..." << endl;

    lib.returnBack(); // Return the book


    cout << "\nAttempting to return the book again..." << endl;

    lib.returnBack(); // Try to return a book that was already returned


    return 0;
}
```
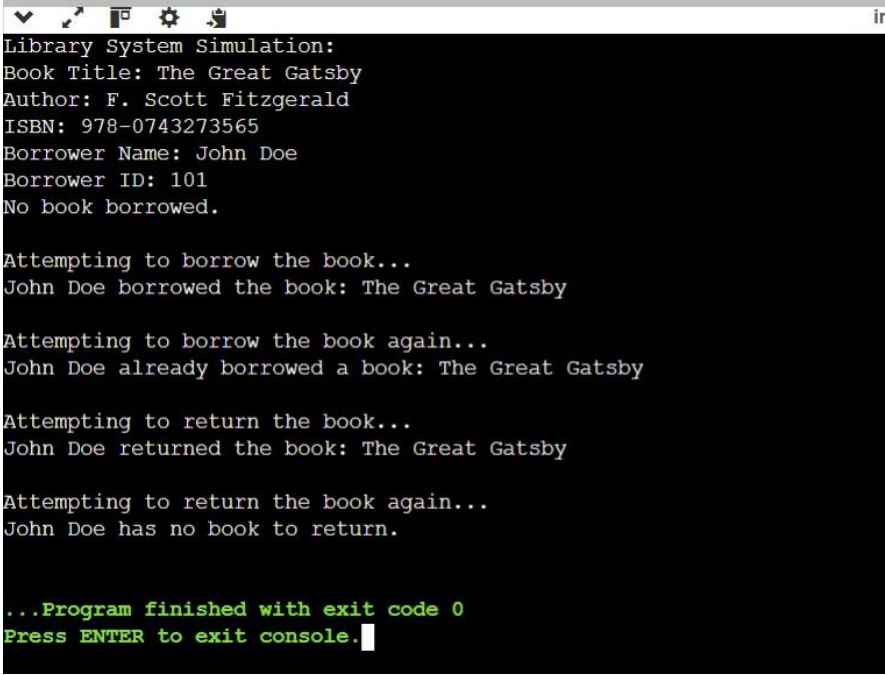
```
Library System Simulation:
Book Title: The Great Gatsby
Author: F. Scott Fitzgerald
ISBN: 978-0743273565
Borrower Name: John Doe
Borrower ID: 101
No book borrowed.

Attempting to borrow the book...
John Doe borrowed the book: The Great Gatsby

Attempting to borrow the book again...
John Doe already borrowed a book: The Great Gatsby

Attempting to return the book...
John Doe returned the book: The Great Gatsby

Attempting to return the book again...
John Doe has no book to return.


...Program finished with exit code 0
Press ENTER to exit console.
```

18. 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.

CODE –

```cpp
#include <iostream>

using namespace std;


// Base class for account

class Account {

protected:

    double balance;

    double rate;  // Interest rate

    double time;  // Time period for interest calculation


public:

    // Constructor to initialize balance, rate, and time

    Account(double bal, double r, double t) : balance(bal), rate(r), time(t) {}


    // Virtual function to calculate interest

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

    virtual void displayAccountDetails() = 0;  // To display account details

};
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
// Derived class for SavingsAccount

class SavingsAccount : public Account {

public:

    // Constructor to initialize SavingsAccount details

    SavingsAccount(double bal, double r, double t) : Account(bal, r, t) {}


    // Overriding the calculateInterest function for SavingsAccount

    void calculateInterest() override {

        double interest = balance * rate * time;

        balance += interest;

        cout << "Interest for Savings Account: " << interest << endl;

    }


    // Function to display details of the Savings Account

    void displayAccountDetails() override {

        cout << "Savings Account Balance: " << balance << endl;

        cout << "Interest Rate: " << rate << "%" << endl;

        cout << "Time Period: " << time << " years" << endl;

    }

};


// Derived class for CurrentAccount

class CurrentAccount : public Account {

private:

    double maintenanceFee;  // Monthly maintenance fee


public:
```

```cpp
    // Constructor to initialize CurrentAccount details

    CurrentAccount(double bal, double r, double t, double fee) : Account(bal, r, t),
maintenanceFee(fee) {}


    // Overriding the calculateInterest function for CurrentAccount

    void calculateInterest() override {

        // No interest for CurrentAccount, but deduct maintenance fee

        balance -= maintenanceFee;

        cout << "Maintenance Fee Deducted: " << maintenanceFee << endl;

    }


    // Function to display details of the Current Account

    void displayAccountDetails() override {

        cout << "Current Account Balance: " << balance << endl;

        cout << "Maintenance Fee: " << maintenanceFee << endl;

    }
};


int main() {

    // Creating a SavingsAccount object

    Account* account1 = new SavingsAccount(1000.0, 0.05, 1);  // Balance = 1000, Rate =
5%, Time = 1 year

    account1->displayAccountDetails();

    account1->calculateInterest();

    cout << "Updated Savings Account Balance: " << account1->balance << endl;


    cout << endl;
```

```
    // Creating a CurrentAccount object

    Account* account2 = new CurrentAccount(1000.0, 0.0, 1, 50);  // Balance = 1000, No
    interest, Maintenance Fee = 50

    account2->displayAccountDetails();

    account2->calculateInterest();

    cout << "Updated Current Account Balance: " << account2->balance << endl;


    // Clean up memory

    delete account1;

    delete account2;


    return 0;

}
```

19. 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.

CODE –

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

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
// Base class Employee
class Employee {
protected:
    string name;

    int id;

    double salary;


public:
    // Constructor to initialize Employee details
    Employee(string n, int i, double s) : name(n), id(i), salary(s) {}


    // Virtual function to calculate total earnings
    virtual void calculateEarnings() = 0;


    // Function to display employee details
    void displayDetails() {
        cout << "Employee Name: " << name << endl;

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

        cout << "Basic Salary: $" << salary << endl;

    }
};


// Derived class Manager
class Manager : public Employee {
private:
    double performanceRating; // Manager's performance rating
```

```cpp
public:
    // Constructor to initialize Manager details
    Manager(string n, int i, double s, double rating)
        : Employee(n, i, s), performanceRating(rating) {}

    // Overriding the calculateEarnings function for Manager
    void calculateEarnings() override {
        double bonus = salary * (performanceRating / 100);  // Bonus based on performance rating
        double totalEarnings = salary + bonus;
        cout << "Performance Rating: " << performanceRating << "%" << endl;
        cout << "Manager's Bonus: $" << bonus << endl;
        cout << "Total Earnings: $" << totalEarnings << endl;
    }
};


// Derived class Developer
class Developer : public Employee {
private:
    int overtimeHours;  // Number of overtime hours worked
    double overtimeRate; // Overtime compensation rate per hour

public:
    // Constructor to initialize Developer details
    Developer(string n, int i, double s, int hours, double rate)
        : Employee(n, i, s), overtimeHours(hours), overtimeRate(rate) {}
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
    // Overriding the calculateEarnings function for Developer

    void calculateEarnings() override {

        double overtimeCompensation = overtimeHours * overtimeRate;  // Compensation for overtime

        double totalEarnings = salary + overtimeCompensation;

        cout << "Overtime Hours Worked: " << overtimeHours << endl;

        cout << "Overtime Compensation: $" << overtimeCompensation << endl;

        cout << "Total Earnings: $" << totalEarnings << endl;

    }

};


int main() {

    string name;

    int id;

    double salary;


    // Input for Manager

    cout << "Enter Manager's Details:" << endl;

    cout << "Name: ";

    getline(cin, name);

    cout << "ID: ";

    cin >> id;

    cout << "Basic Salary: $";

    cin >> salary;

    double rating;

    cout << "Performance Rating (%): ";

    cin >> rating;
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
Manager manager(name, id, salary, rating);

manager.displayDetails();

manager.calculateEarnings();


cout << endl;


// Input for Developer

cin.ignore();  // To clear the input buffer from the previous cin

cout << "Enter Developer's Details:" << endl;

cout << "Name: ";

getline(cin, name);

cout << "ID: ";

cin >> id;

cout << "Basic Salary: $";

cin >> salary;

int hours;

double rate;

cout << "Overtime Hours Worked: ";

cin >> hours;

cout << "Overtime Rate ($ per hour): ";

cin >> rate;


Developer developer(name, id, salary, hours, rate);

developer.displayDetails();

developer.calculateEarnings();
```
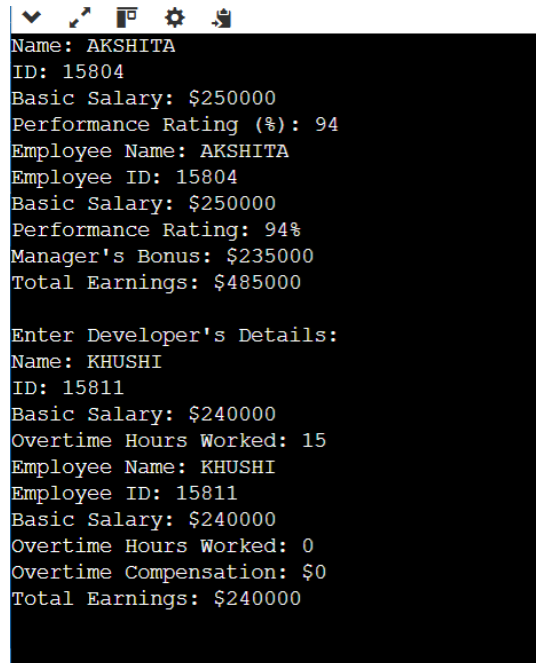
NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```
    return 0;

}
```



```
Name: AKSHITA
ID: 15804
Basic Salary: $250000
Performance Rating (%): 94
Employee Name: AKSHITA
Employee ID: 15804
Basic Salary: $250000
Performance Rating: 94%
Manager's Bonus: $235000
Total Earnings: $485000

Enter Developer's Details:
Name: KHUSHI
ID: 15811
Basic Salary: $240000
Overtime Hours Worked: 15
Employee Name: KHUSHI
Employee ID: 15811
Basic Salary: $240000
Overtime Hours Worked: 0
Overtime Compensation: $0
Total Earnings: $240000
```

20. Create a C++ program to simulate a vehicle hierarchy using multi-level inheritance. Design a base class Vehicle that stores basic details (brand, model, and mileage). Extend it into the Car class to add attributes like fuel efficiency and speed. Further extend it into ElectricCar to include battery capacity and charging time. Implement methods to calculate:

Fuel Efficiency: Miles per gallon (for Car).

Range: Total distance the electric car can travel with a full charge.

CODE -

#include <iostream>

#include <string>

using namespace std;

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
// Base class Vehicle
class Vehicle {
protected:
    string brand;
    string model;
    double mileage; // Miles per gallon for fuel-based vehicles

public:
    // Constructor to initialize Vehicle details
    Vehicle(string b, string m, double mi) : brand(b), model(m), mileage(mi) {}

    // Method to display vehicle details
    void displayDetails() {
        cout << "Brand: " << brand << endl;
        cout << "Model: " << model << endl;
        cout << "Mileage: " << mileage << " miles per gallon" << endl;
    }
};

// Derived class Car extends Vehicle
class Car : public Vehicle {
protected:
    double fuelEfficiency; // Miles per gallon
    double speed; // Speed in miles per hour

public:
    // Constructor to initialize Car details
```

```cpp
Car(string b, string m, double mi, double fe, double s)
    : Vehicle(b, m, mi), fuelEfficiency(fe), speed(s) {}


// Method to calculate fuel efficiency
void calculateFuelEfficiency() {

    cout << "Fuel Efficiency: " << fuelEfficiency << " miles per gallon" << endl;

}


// Method to display car-specific details
void displayCarDetails() {

    displayDetails();

    calculateFuelEfficiency();

    cout << "Speed: " << speed << " miles per hour" << endl;

}
};


// Derived class ElectricCar extends Car
class ElectricCar : public Car {
private:
    double batteryCapacity; // in kWh

    double chargingTime;    // in hours


public:
    // Constructor to initialize ElectricCar details
    ElectricCar(string b, string m, double mi, double fe, double s, double bc, double ct)
        : Car(b, m, mi, fe, s), batteryCapacity(bc), chargingTime(ct) {}
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
    // Method to calculate the range of the electric car

    void calculateRange() {

        double range = batteryCapacity * 4; // Assuming each kWh gives 4 miles of range
(example)

        cout << "Range on full charge: " << range << " miles" << endl;

    }


    // Method to display electric car-specific details

    void displayElectricCarDetails() {

        displayCarDetails();

        calculateRange();

        cout << "Battery Capacity: " << batteryCapacity << " kWh" << endl;

        cout << "Charging Time: " << chargingTime << " hours" << endl;

    }
};

int main() {
    // Creating an ElectricCar object
    ElectricCar myElectricCar("Tesla", "Model S", 100, 120, 150, 85, 1.5);


    // Displaying details of the electric car
    cout << "Electric Car Details:" << endl;
    myElectricCar.displayElectricCarDetails();


    return 0;

}
```
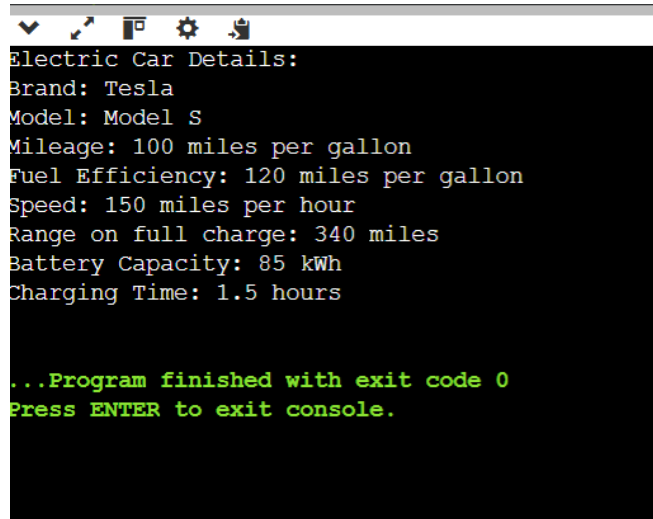
```
Electric Car Details:
Brand: Tesla
Model: Model S
Mileage: 100 miles per gallon
Fuel Efficiency: 120 miles per gallon
Speed: 150 miles per hour
Range on full charge: 340 miles
Battery Capacity: 85 kWh
Charging Time: 1.5 hours


...Program finished with exit code 0
Press ENTER to exit console.
```

21. Design a C++ program using **function overloading** to perform arithmetic operations on complex numbers. Define a Complex class with real and imaginary parts. Overload functions to handle the following operations:

Addition: Sum of two complex numbers.

Multiplication: Product of two complex numbers.

Magnitude: Calculate the magnitude of a single complex number.

The program should allow the user to select an operation, input complex numbers, and display results in the format a + bi or a - bi (where b is the imaginary part).

CODE –

```cpp
#include <iostream>
#include <cmath> // For sqrt() function to calculate magnitude
using namespace std;

class Complex {
private:
    double real, imag;
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
public:
    // Constructor to initialize complex number
    Complex(double r = 0, double i = 0) : real(r), imag(i) {}

    // Function to display complex number in the format a + bi or a - bi
    void display() {
        if (imag >= 0)
            cout << real << " + " << imag << "i" << endl;
        else
            cout << real << " - " << -imag << "i" << endl;
    }

    // Overloading + operator for addition of two complex numbers
    Complex operator+(const Complex& other) {
        return Complex(real + other.real, imag + other.imag);
    }

    // Overloading * operator for multiplication of two complex numbers
    Complex operator*(const Complex& other) {
        return Complex(real * other.real - imag * other.imag,
                real * other.imag + imag * other.real);
    }

    // Function to calculate the magnitude (modulus) of a complex number
    double magnitude() {
        return sqrt(real * real + imag * imag);
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
    }
};


int main() {
    double r1, i1, r2, i2;

    int choice;


    // Input complex number 1
    cout << "Enter real and imaginary part of first complex number: ";

    cin >> r1 >> i1;

    Complex c1(r1, i1);


    // Input complex number 2
    cout << "Enter real and imaginary part of second complex number: ";

    cin >> r2 >> i2;

    Complex c2(r2, i2);


    // Menu for operation selection
    cout << "Choose the operation:\n";

    cout << "1. Add\n2. Multiply\n3. Magnitude of first complex number\n4. Magnitude of second complex number\n";

    cin >> choice;


    switch (choice) {
        case 1: {

            Complex result = c1 + c2; // Addition of complex numbers

            cout << "The sum of the complex numbers is: ";
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
        result.display();

        break;

    }

    case 2: {

        Complex result = c1 * c2; // Multiplication of complex numbers

        cout << "The product of the complex numbers is: ";

        result.display();

        break;

    }

    case 3: {

        double result = c1.magnitude(); // Magnitude of the first complex number

        cout << "The magnitude of the first complex number is: " << result << endl;

        break;

    }

    case 4: {

        double result = c2.magnitude(); // Magnitude of the second complex number

        cout << "The magnitude of the second complex number is: " << result << endl;

        break;

    }

    default:

        cout << "Invalid choice!" << endl;

    }


    return 0;

 }
```
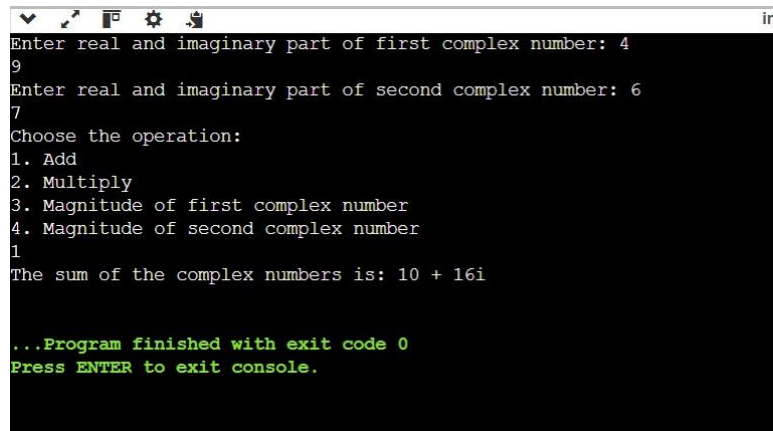
NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```
Enter real and imaginary part of first complex number: 4
9
Enter real and imaginary part of second complex number: 6
7
Choose the operation:
1. Add
2. Multiply
3. Magnitude of first complex number
4. Magnitude of second complex number
1
The sum of the complex numbers is: 10 + 16i


...Program finished with exit code 0
Press ENTER to exit console.
```

22. Create a C++ program that uses polymorphism to calculate the area of various shapes. Define a base class Shape with a virtual method calculateArea(). Extend this base class into the following derived classes:

Rectangle: Calculates the area based on length and width.

Circle: Calculates the area based on the radius.

Triangle: Calculates the area using base and height.

The program should use dynamic polymorphism to handle these shapes and display the area of each.

CODE –

#include <iostream>

#include <cmath> // For M_PI to get the value of pi

using namespace std;

// Base class Shape

class Shape {

public:

    // Virtual function to calculate the area

```cpp
    virtual double calculateArea() = 0; // Pure virtual function

};


// Derived class Rectangle

class Rectangle : public Shape {

private:

    double length, width;


public:

    // Constructor to initialize length and width

    Rectangle(double l, double w) : length(l), width(w) {}


    // Override the calculateArea method

    double calculateArea() override {

        return length * width; // Area of rectangle: length * width

    }

};


// Derived class Circle

class Circle : public Shape {

private:

    double radius;


public:

    // Constructor to initialize radius

    Circle(double r) : radius(r) {}
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
    // Override the calculateArea method

    double calculateArea() override {

        return M_PI * radius * radius; // Area of circle: π * r^2

    }

};


// Derived class Triangle

class Triangle : public Shape {

private:

    double base, height;


public:

    // Constructor to initialize base and height

    Triangle(double b, double h) : base(b), height(h) {}


    // Override the calculateArea method

    double calculateArea() override {

        return 0.5 * base * height; // Area of triangle: 1/2 * base * height

    }

};


// Function to display the area of the shape

void displayArea(Shape* shape) {

    cout << "Area: " << shape->calculateArea() << endl;

}


int main() {
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
// Create instances of different shapes
Rectangle rect(10.0, 5.0);

Circle circ(7.0);

Triangle tri(6.0, 8.0);


// Use dynamic polymorphism to display the area of each shape
cout << "Rectangle: ";

displayArea(&rect); // Calls Rectangle's calculateArea


cout << "Circle: ";

displayArea(&circ); // Calls Circle's calculateArea


cout << "Triangle: ";

displayArea(&tri); // Calls Triangle's calculateArea


return 0;
}
```
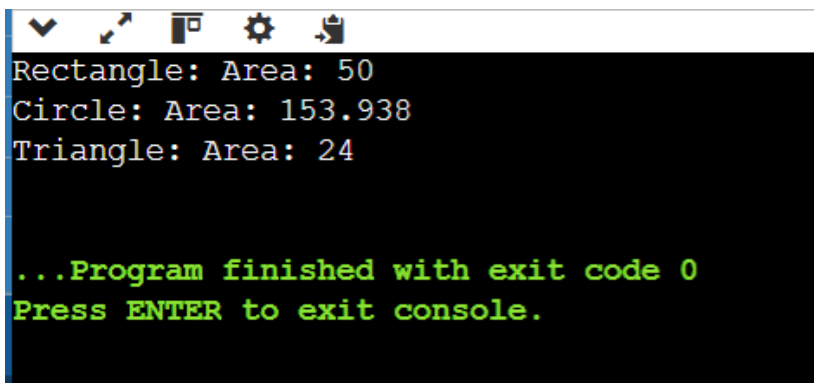
```
Rectangle: Area: 50
Circle: Area: 153.938
Triangle: Area: 24


...Program finished with exit code 0
Press ENTER to exit console.
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

23. 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.

CODE –

```cpp
#include <iostream>
#include <cmath> // For M_PI to get the value of Pi
using namespace std;

// Overloaded function to calculate the area of a circle
double calculateArea(double radius) {
    if (radius <= 0) {
        cout << "Invalid input! Radius must be greater than 0." << endl;
        return -1;
    }
    return M_PI * radius * radius; // Area of circle: π * r^2
}

// Overloaded 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 greater than 0." << endl;
        return -1;
```

```cpp
    }
    return length * breadth; // Area of rectangle: length * breadth
}

// Overloaded 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 greater than 0." << endl;
        return -1;
    }
    return 0.5 * base * height; // Area of triangle: 1/2 * base * height
}

int main() {
    int choice;
    double area;

    cout << "Geometric Shape Area Calculator\n";
    cout << "1. Circle\n";
    cout << "2. Rectangle\n";
    cout << "3. Triangle\n";
    cout << "Enter your choice (1-3): ";
    cin >> choice;

    switch (choice) {
        case 1: {
            double radius;
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

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

        cin >> radius;

        area = calculateArea(radius);

        if (area != -1) {

            cout << "Area of the circle: " << area << endl;

        }

        break;

    }

    case 2: {

        double length, breadth;

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

        cin >> length >> breadth;

        area = calculateArea(length, breadth);

        if (area != -1) {

            cout << "Area of the rectangle: " << area << endl;

        }

        break;

    }

    case 3: {

        double base, height;

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

        cin >> base >> height;

        area = calculateArea(base, height);

        if (area != -1) {

            cout << "Area of the triangle: " << area << endl;

        }

        break;
```

NAME –Ankur Chauhan
UID – 22BCS14662
SECTION – FL-IOT-603-B

```cpp
        }

        default:

            cout << "Invalid choice! Please select a valid option (1-3)." << endl;

            break;

    }


    return 0;

}
```



```
Geometric Shape Area Calculator
1. Circle
2. Rectangle
3. Triangle
Enter your choice (1-3): 1
Enter radius of the circle: 5
Area of the circle: 78.5398

...Program finished with exit code 0
Press ENTER to exit console.
```