DAY : 1

Student Name : Piyush Bhardwaj

U : 22BCS11108

Course : BE – CSE

Section/Group : IOT-639-B

Semester : 5th

Datof Performance : 19-12-2024

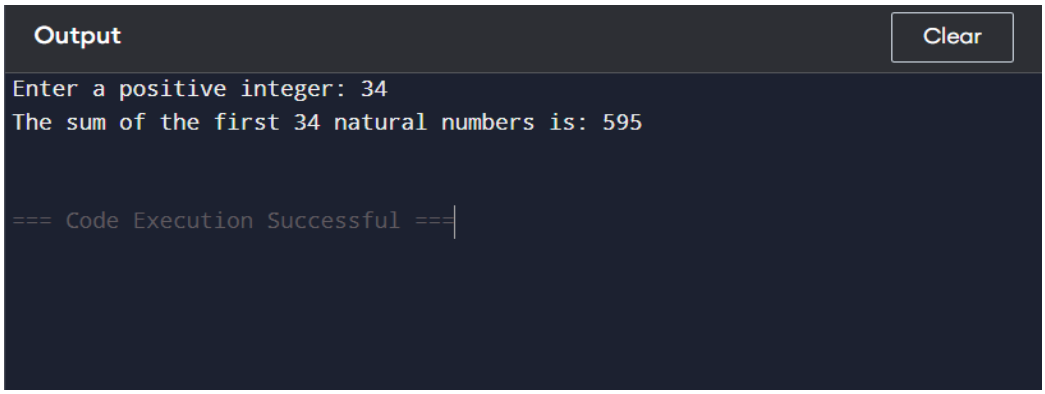
Subject Name : C++ and DSA

Program 1 : Write a C++ program to calculate the total sum of first n natural numbers

Input:

```
#include <iostream>

using namespace std;

int main() {

  int n, sum = 0;

  cout << "Enter a positive integer: ";

  cin >> n;

  for (int i = 1; i <= n; ++i) {

    sum += i;

  }

  cout << "The sum of the first " << n << " natural numbers is: " << sum << endl;

  return 0;

}
```

Output :



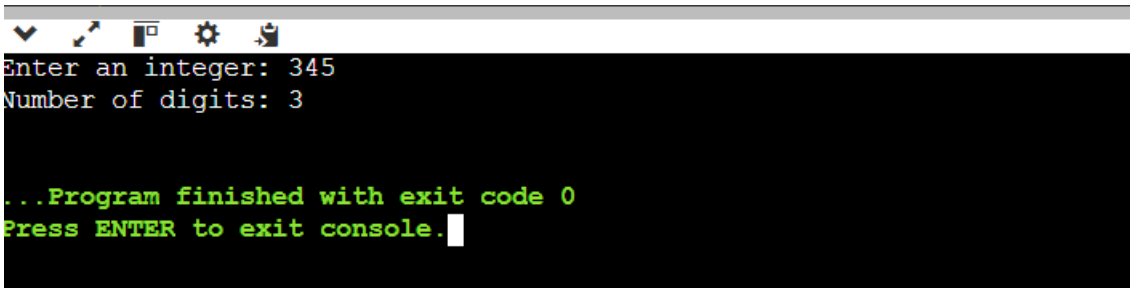
Program 2 : Write a C++ program to count the number of digits in an integer.

Input:

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

int main() {
  int number, count = 0;

  // Input from the user
  cout << "Enter an integer: ";
  cin >> number;

  // Handle negative numbers
  if (number < 0) {
    number = -number;
  }

  // Special case for zero
  if (number == 0) {
    count = 1;
  } else {
    // Count digits
    while (number != 0) {
      number /= 10;
      count++;
    }
  }
```

```
    // Display the result
    cout << "Number of digits: " << count << endl;

    return 0;
}
```

Output:



Program 3 : Write a C++ program to find the largest digit in a given number:

Input :

```
#include <iostream>
using namespace std
int main() {
    int number, largestDigit = 0;
    // Input from the user
    cout << "Enter an integer: ";
    cin >> number;

    // Handle negative numbers
    if (number < 0) {
        number = -number;
    }

    // Find the largest digit
```

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

    // Display the result
    cout << "The largest digit is: " << largestDigit << endl;

    return 0;
}
```

Output:

```
Enter an integer: 9876789
The largest digit is: 9


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

VERY EASY PROBLEM :

Program 4 : Write a C++ program to check whether a number is prime.

Input:

```
#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;
    }

    return true;
}

int main() {
    int number;

    // Input from the user
    cout << "Enter an integer: ";
    cin >> number;
    // Check if the number is prime
    if (isPrime(number))
        cout << number << " is a prime number." << endl;
    else
        cout << number << " is not a prime number." << endl;
    return 0;
}
```

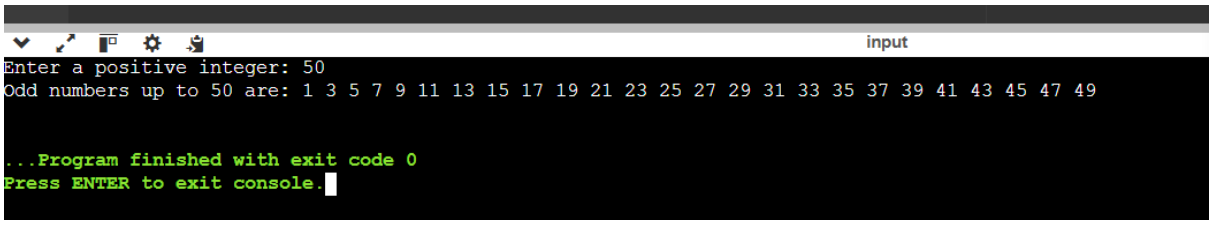
Output:

Program 5 : Write a C++ program to print all odd numbers .

Input:

```
#include <iostream>

using namespace std;

int main() {

   int n;


   // Input from the user

   cout << "Enter a positive integer: ";

   cin >> n;


   cout << "Odd numbers up to " << n << " are: ";


   // Loop to print odd numbers

   for (int i = 1; i <= n; i += 2) {

      cout << i << " ";

   }


   cout << endl;


   return 0;

}
```

Output:



EASY PROBLEM :

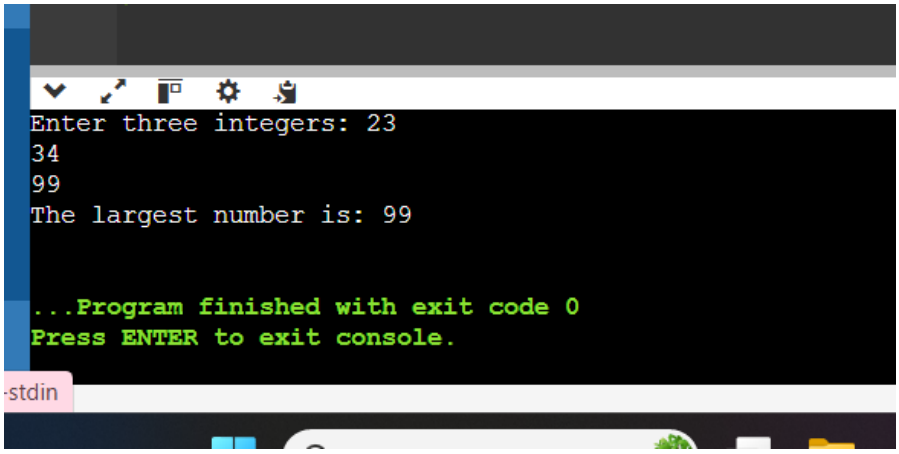
Program 6 : Write a C++ program to find the largest number .

Input :

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

int main() {
   int num1, num2, num3;

   // Input from the user
   cout << "Enter three integers: ";
   cin >> num1 >> num2 >> num3;

   // Find the largest number
   int largest = num1;

   if (num2 > largest) {
      largest = num2;
   }
   if (num3 > largest) {
      largest = num3;
   }

   // Display the result
   cout << "The largest number is: " << largest << endl;

   return 0;
```

}

Output :



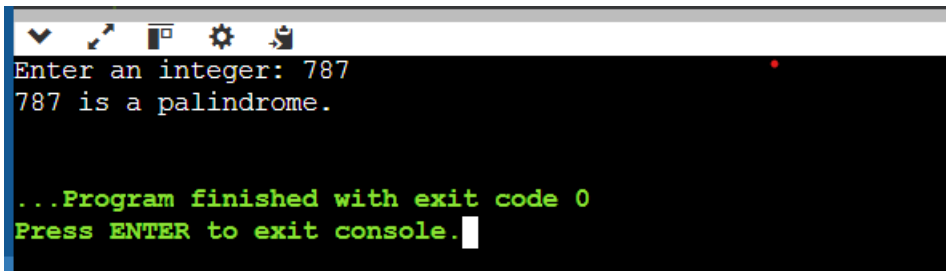
Program 7 : Write a C++ program to check if a number is a palindrome.

Input :

```
#include <iostream>

using namespace std;


int main() {

   int number, originalNumber, reversedNumber = 0, remainder;


   // Input from the user

   cout << "Enter an integer: ";

   cin >> number;


   originalNumber = number;


   // Reverse the number

   while (number != 0) {

      remainder = number % 10;

      reversedNumber = reversedNumber * 10 + remainder;
```

```
    number /= 10;
  }

  // Check if the original number is equal to the reversed number
  if (originalNumber == reversedNumber)
    cout << originalNumber << " is a palindrome." << endl;
  else
    cout << originalNumber << " is not a palindrome." << endl;

  return 0;
}
```

Output:



MEDIUM PROBLEM :

Program 8 : Write a program in C++ Encapsulation using Employee Details .

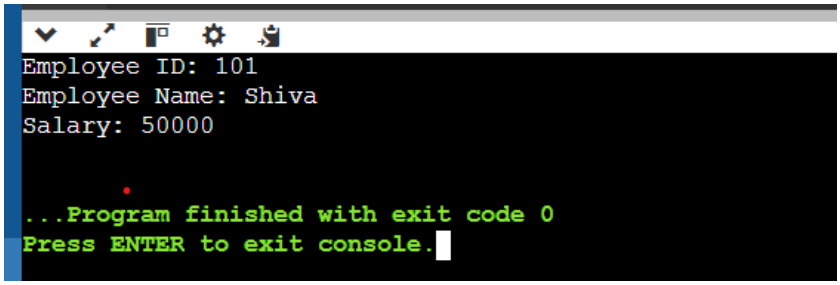
Input :

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

// Employee class with private data members
class Employee {
private:
```

```
    int employeeID;
    string employeeName;
    double salary;

public:
    // Setter function for employee ID
    void setEmployeeID(int id) {
        employeeID = id;
    }

    // Getter function for employee ID
    int getEmployeeID() {
        return employeeID;
    }

    // Setter function for employee name
    void setEmployeeName(string name) {
        employeeName = name;
    }

    // Getter function for employee name
    string getEmployeeName() {
        return employeeName;
    }

    // Setter function for salary
    void setSalary(double s) {
        salary = s;
```

```
    }

    // Getter function for salary
    double getSalary() {
        return salary;
    }
    void displayDetails() {
        cout << "Employee ID: " << getEmployeeID() << endl;
        cout << "Employee Name: " << getEmployeeName() << endl;
        cout << "Salary: " << getSalary() << endl;
    }
};

int main() {
    Employee emp;
    emp.setEmployeeID(101);
    emp.setEmployeeName("John Doe");
    emp.setSalary(50000.0);
    emp.displayDetails();
    return 0;
}
```

Output :



Program 9 : Write a program in C++ for Inheritance using student's result of class.

Input :

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

// Base class Student
class Student {
protected:
   string name;
   int rollNumber;

public:
   // Setter methods for student details
   void setStudentDetails(string studentName, int studentRollNumber) {
      name = studentName;
      rollNumber = studentRollNumber;
   }

   // Getter methods for student details
   void displayStudentDetails() {
      cout << "Name: " << name << endl;
      cout << "Roll Number: " << rollNumber << endl;
   }
};

// Derived class Result
class Result : public Student {
private:
```

```
    int marks[5];
    float percentage;

public:
    // Setter method for marks
    void setMarks(int m1, int m2, int m3, int m4, int m5) {
        marks[0] = m1;
        marks[1] = m2;
        marks[2] = m3;
        marks[3] = m4;
        marks[4] = m5;
    }

    // Function to calculate the percentage
    void calculatePercentage() {
        int totalMarks = 0;
        for (int i = 0; i < 5; i++) {
            totalMarks += marks[i];
        }
        percentage = (float)totalMarks / 5;
    }
    void displayResult() {
        displayStudentDetails();
        cout << "Marks obtained in 5 subjects: ";
        for (int i = 0; i < 5; i++) {
            cout << marks[i] << " ";
        }
        cout << endl;
```

```
        cout << "Percentage: " << percentage << "%" << endl;
        if (percentage >= 50) {
            cout << "Result: Passed" << endl;
        } else {
            cout << "Result: Failed" << endl;
        }
    }
};
int main() {
    Result student1;
    student1.setStudentDetails("KRISHNA", 101);
    student1.setMarks(85, 90, 78, 88, 92);
    student1.calculatePercentage();
    student1.displayResult();
    return 0;
}
```

Output :

```
Name: KRISHNA
Roll Number: 101
Marks obtained in 5 subjects: 85 90 78 88 92
Percentage: 86.6%
Result: Passed
```

HARD PROBLEM :

PROGRAM 10 : Write a program in C++ for Matrix using function overloading.

Input:

```
#include <iostream>
using namespace std;
class Matrix {
private:
  int mat[2][2];  // 2x2 matrix for simplicity
public:
  void inputMatrix() {
    cout << "Enter elements of the matrix (2x2): " << endl;
    for (int i = 0; i < 2; i++) {
      for (int j = 0; j < 2; j++) {
        cin >> mat[i][j];
      }
    }
  }
  void displayMatrix() {
    cout << "Matrix: " << endl;
    for (int i = 0; i < 2; i++) {
      for (int j = 0; j < 2; j++) {
        cout << mat[i][j] << " ";
      }
      cout << endl;
    }
  }
```

```
    // Function to add two matrices (Function Overloading)
    void addMatrices(Matrix m) {
        cout << "Matrix after addition: " << endl;
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                mat[i][j] += m.mat[i][j];
            }
        }
        displayMatrix();
    }
    void subtractMatrices(Matrix m) {
        cout << "Matrix after subtraction: " << endl;
        for (int i = 0; i < 2; i++) {
            for (int j = 0; j < 2; j++) {
                mat[i][j] -= m.mat[i][j];
            }
        }
        displayMatrix();
    }
};
int main() {
    Matrix m1, m2;
    cout << "Enter the first matrix:" << endl;
    m1.inputMatrix();
    cout << "Enter the second matrix:" << endl;
    m2.inputMatrix();
    cout << "\nFirst Matrix: " << endl;
    m1.displayMatrix();
```

```
    cout << "\nSecond Matrix: " << endl;

    m2.displayMatrix();

    m1.addMatrices(m2);

    m1.subtractMatrices(m2);

    return 0;

}
```

Output:

```
input
Enter the first matrix:
Enter elements of the matrix (2x2):
1
2
3
4
Enter the second matrix:
Enter elements of the matrix (2x2):
5
6
7
8

First Matrix:
Matrix:
1 2
3 4

Second Matrix:
Matrix:
5 6
7 8
Matrix after addition:
Matrix:
6 8
10 12
Matrix after subtraction:
Matrix:
1 2
3 4


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

Program 11 : Write a program in C++ for Polymorphism in shape classes.

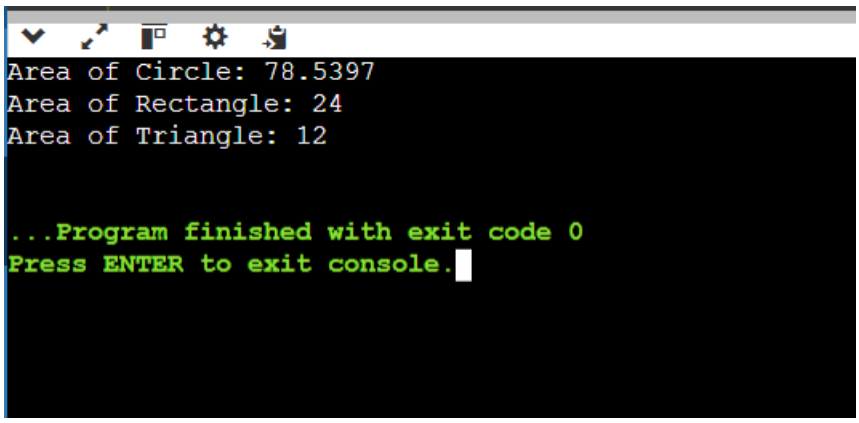
Input :

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

// Base class Shape
class Shape {
public:
   // Virtual function for calculating area
   virtual void area() {
      cout << "Calculating area of a shape..." << endl;
   }
};

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

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

   // Overriding the area function for Circle
   void area() override {
      float area = 3.14159 * radius * radius;
      cout << "Area of Circle: " << area << endl;
   }
```

```
};

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

public:
   Rectangle(float l, float w) : length(l), width(w) {}

   // Overriding the area function for Rectangle
   void area() override {
      float area = length * width;
      cout << "Area of Rectangle: " << area << endl;
   }
};

// Derived class Triangle
class Triangle : public Shape {
private:
   float base, height;

public:
   Triangle(float b, float h) : base(b), height(h) {}

   // Overriding the area function for Triangle
   void area() override {
      float area = 0.5 * base * height;
```

```
    cout << "Area of Triangle: " << area << endl;
  }
};

int main() {
  // Creating objects of different shapes
  Shape* shape1 = new Circle(5.0);  // Circle with radius 5.0
  Shape* shape2 = new Rectangle(4.0, 6.0);  // Rectangle with length 4.0 and width 6.0
  Shape* shape3 = new Triangle(4.0, 6.0);  // Triangle with base 4.0 and height 6.0

  // Calling the area function on different shapes using polymorphism
  shape1->area();  // Circle's area
  shape2->area();  // Rectangle's area
  shape3->area();  // Triangle's area
  delete shape1;
  delete shape2;
  delete shape3;
  return 0;
}
```

Output :



VERY HARD PROBLEM :

Program 12 : Write a program in C++ for Advanced function overloading for geometric shapes objectives .

Input :

```
#include <iostream>

#include <cmath>

using namespace std;


// Base class Shape

class Shape {

public:

   // Virtual function to calculate area

   virtual void calculate() = 0;

};


// Derived class Circle

class Circle : public Shape {

private:

   float radius;


public:

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


   // Overloading calculate() to calculate area and perimeter for Circle

   void calculate() override {

      float area = M_PI * radius * radius;

      float perimeter = 2 * M_PI * radius;
```

```
        cout << "Circle: Area = " << area << ", Perimeter = " << perimeter << endl;
    }
};

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

public:
    Rectangle(float l, float w) : length(l), width(w) {}

    // Overloading calculate() to calculate area and perimeter for Rectangle
    void calculate() override {
        float area = length * width;
        float perimeter = 2 * (length + width);
        cout << "Rectangle: Area = " << area << ", Perimeter = " << perimeter << endl;
    }
};

// Derived class Triangle
class Triangle : public Shape {
private:
    float base, height, side1, side2, side3;

public:
    Triangle(float b, float h, float s1, float s2, float s3)
        : base(b), height(h), side1(s1), side2(s2), side3(s3) {}
```

```
    // Overloading calculate() to calculate area and perimeter for Triangle
    void calculate() override {
        float area = 0.5 * base * height;
        float perimeter = side1 + side2 + side3;
        cout << "Triangle: Area = " << area << ", Perimeter = " << perimeter << endl;
    }
};

// Advanced Function Overloading: Calculate using different parameters
class ShapeCalculator {
public:
    // Overloaded calculate function for Circle (single parameter)
    void calculate(float radius) {
        Circle circle(radius);
        circle.calculate();
    }

    // Overloaded calculate function for Rectangle (two parameters)
    void calculate(float length, float width) {
        Rectangle rectangle(length, width);
        rectangle.calculate();
    }

    // Overloaded calculate function for Triangle (five parameters)
    void calculate(float base, float height, float side1, float side2, float side3) {
        Triangle triangle(base, height, side1, side2, side3);
        triangle.calculate();
```

```
    }
};

int main() {
    ShapeCalculator calculator;

    // Calculate for different shapes using overloaded calculate function
    calculator.calculate(5.0f);  // Circle with radius 5.0
    calculator.calculate(4.0f, 6.0f);  // Rectangle with length 4.0 and width 6.0
    calculator.calculate(3.0f, 4.0f, 3.0f, 4.0f, 5.0f);  // Triangle with base 3.0, height 4.0, sides 3.0, 4.0, 5.0

    return 0;
}
```

Output :

```
Circle: Area = 78.5398, Perimeter = 31.4159
Rectangle: Area = 24, Perimeter = 20
Triangle: Area = 6, Perimeter = 12


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