

Assignment 1

Name – Ishita

Faculty Name- Er. Rajni Devi

UID- 22BCS15353

Date- 19 Dec ,2024

Section – 620-B

Very Easy

Ques 1:

Calculate the sum of all natural numbers from 1 to n, where n is a positive integer. Use the formula:

$$\text{Sum} = n \times (n+1) / 2 .$$

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;

    cout<<"enter a number -";

    cin>>n;

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

    cout<< result;

    return 0;

}
```

Output:

```
Output
enter a number -5
15

=== Code Execution Successful ===
```

Ques 2:

Check if a given number n is a prime number. Given an integer n, print "Prime" if the number is prime, or "Not Prime" if it is not.

Code:

```
#include <iostream>

using namespace std;

int main() {
    int count=0,n;
    cout<<"enter a number - ";
    cin>>n;
    for(int i =1;i<=n;i++){
        if(n%i==0)
            count++;
    }
    if(count==2)
        cout<<"Prime Number";
    else
        cout<<"Not a Prime Number";
    return 0;
}
```

Output:

Output

```
enter a number - 7
Prime Number
```

```
=== Code Execution Successful ===
```

Easy

Ques 3:

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>

using namespace std;

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

    cout << "Enter a Number - ";

    cin >> n;

    do {
        n /= 10;
        count++;
    } while (n != 0);

    cout << "Number of digits: " << count;

    return 0;
}
```

Output:

Output

Enter a Number - 12345

Number of digits: 5

=== Code Execution Successful ===

Ques 4:

Check whether a given number is a palindrome or not. Given an integer n, print "Palindrome" if the number is a palindrome, otherwise print "Not Palindrome".

Code:

```
#include <iostream>

using namespace std;

int main() {

    int n,rem,rev=0,org;

    cout<<"enter a number ";

    cin>>n;

    org=n;

    while(n!=0){

        rem=n%10;

        rev=rev*10+rem;

        n/=10;

    }

    if(org==rev)

        cout<<"Palindrome Number";

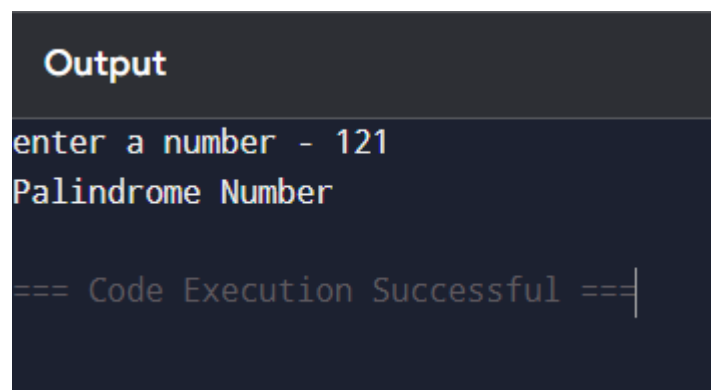
    else

        cout<<"Not a Palindrome Number";

    return 0;

}
```

Output:



The screenshot shows a dark-themed terminal window. At the top, the word "Output" is displayed in a light blue font. Below it, the program's output is shown in a light green monospace font: "enter a number - 121" followed by "Palindrome Number" on the next line. At the bottom of the window, a status message in a light gray font reads "=== Code Execution Successful ===" followed by a vertical cursor bar.

Medium

Ques 5:

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;

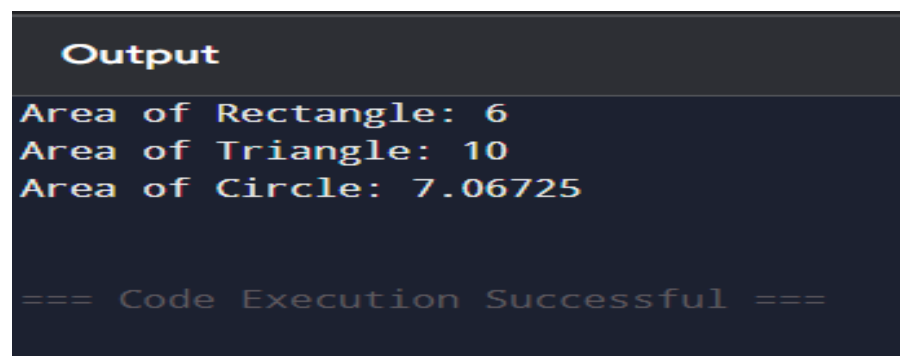
void area(int l, int b) {
    cout << "Area of Rectangle: " << l * b << endl;
}

void area(int b, float h) {
    cout << "Area of Triangle: " << 0.5 * b * h << endl;
}

void area(float r) {
    cout << "Area of Circle: " << 3.141f * r * r << endl;
}

int main() {
    area(2, 3);
    area(4, 5.0f);
    area(1.5f);
    return 0;
}
```

Output:



```
Output
Area of Rectangle: 6
Area of Triangle: 10
Area of Circle: 7.06725

=== Code Execution Successful ===
```

Ques 6:

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:

```
#include <iostream>

#include <iomanip>

using namespace std;

class Circle {
    double radius;

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

    double area() { return 3.14159 * radius * radius; }
};

class Rectangle {
    double length, breadth;

public:
    Rectangle(double l, double b) : length(l), breadth(b) {}

    double area() { return length * breadth; }
};

class Triangle {
    double base, height;

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

    double area() { return 0.5 * base * height; }
};

int main() {
    double radius, length, breadth, base, height;
```

```

cin >> radius >> length >> breadth >> base >> height;

Circle circle(radius);

Rectangle rectangle(length, breadth);

Triangle triangle(base, height);

cout << fixed << setprecision(2);

cout << "Circle Area: " << circle.area() << endl;

cout << "Rectangle Area: " << rectangle.area() << endl;

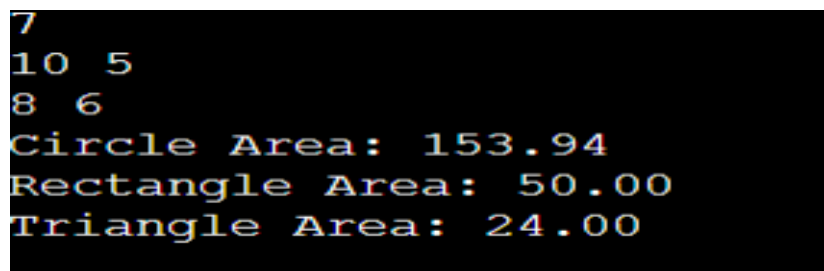
cout << "Triangle Area: " << triangle.area() << endl;

return 0;

}

```

Output:



```

7
10 5
8 6
Circle Area: 153.94
Rectangle Area: 50.00
Triangle Area: 24.00

```

Hard

Ques 7:

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:

```

#include <iostream>

#include <iomanip>

using namespace std;

class Circle {
    double radius;

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

    double area() { return 3.14159 * radius * radius; }
}

```

```

};

class Rectangle {
    double length, breadth;

public:
    Rectangle(double l, double b) : length(l), breadth(b) {}

    double area() { return length * breadth; }
};

class Triangle {
    double base, height;

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

    double area() { return 0.5 * base * height; }
};

int main() {
    double radius, length, breadth, base, height;

    cin >> radius >> length >> breadth >> base >> height;

    Circle circle(radius);

    Rectangle rectangle(length, breadth);

    Triangle triangle(base, height);

    cout << fixed << setprecision(5);

    cout << "Area of Circle: " << circle.area() << endl;

    cout << "Area of Rectangle: " << rectangle.area() << endl;

    cout << "Area of Triangle: " << triangle.area() << endl;

    return 0;}

```

Output:

```

5
4 6
3 7
Area of Circle: 78.53975
Area of Rectangle: 24.00000
Area of Triangle: 10.50000

```


Ques 8:

Design a C++ program using polymorphism to calculate the area of different shapes:

A **Rectangle** (Area = Length \times Breadth).

A **Circle** (Area = $\pi \times \text{Radius}^2$).

A **Triangle** (Area = $\frac{1}{2} \times \text{Base} \times \text{Height}$).

Create a base class Shape with a pure virtual function getArea(). Use derived classes Rectangle, Circle, and Triangle to override this function.

Code:

```
#include <iostream>

#include <iomanip>

using namespace std;

class Shape {
public:
    virtual double getArea() = 0; // Pure virtual function
};

class Rectangle : public Shape {
    int length, breadth;
public:
    Rectangle(int l, int b) : length(l), breadth(b) {}
    double getArea() override { return length * breadth; }
};

class Circle : public Shape {
    int radius;
public:
    Circle(int r) : radius(r) {}
    double getArea() override { return 3.14159 * radius * radius; }
};

class Triangle : public Shape {
    int base, height;
```

```

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

    double getArea() override { return 0.5 * base * height; }
};

int main() {
    int type;

    cin >> type;

    Shape *shape = nullptr; // Pointer to base class

    if (type == 1) { // Rectangle
        int l, b;

        cin >> l >> b;

        shape = new Rectangle(l, b);
    } else if (type == 2) { // Circle
        int r;

        cin >> r;

        shape = new Circle(r);
    } else if (type == 3) { // Triangle
        int b, h;

        cin >> b >> h;

        shape = new Triangle(b, h);
    } else {
        cout << "Invalid shape type";

        return 0;
    }

    cout << fixed << setprecision(2);

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

    delete shape;

    return 0;
}

```

Output:

```
1
5 4
The area of the shape is: 20.00
```

Very Hard

Ques 9:

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 <iomanip>

using namespace std;

class Vehicle {
protected:
    string brand, model;
    double mileage;
public:
    Vehicle(string b, string m, double mi) : brand(b), model(m), mileage(mi) {}

    void displayInfo() {
        cout << "Vehicle: " << brand << " " << model << endl;
        cout << "Mileage: " << mileage << endl;
    }
};
```

```

class Car : public Vehicle {
protected:
    double fuel, distance;
public:
    Car(string b, string m, double mi, double f, double d) : Vehicle(b, m, mi), fuel(f),
distance(d) {}

    double calculateFuelEfficiency() { return distance / fuel; }
};

class ElectricCar : public Car {
    double batteryCapacity, efficiency;
public:
    ElectricCar(string b, string m, double mi, double bc, double eff)
        : Car(b, m, mi, 0, 0), batteryCapacity(bc), efficiency(eff) {}

    double calculateRange() { return batteryCapacity * efficiency; }
};

int main() {
    int type;

    cout << "Vehicle Type (1 for Car, 2 for Electric Car): ";

    cin >> type;

    string brand, model;

    double mileage;

    cout << "Enter Brand, Model, and Mileage: ";

    cin >> brand >> model >> mileage;

    if (type == 1) { // Car
        double fuel, distance;

        cout << "Enter Fuel (gallons) and Distance (miles): ";

        cin >> fuel >> distance;

        Car car(brand, model, mileage, fuel, distance);

        car.displayInfo();
    }
}

```

```

        cout << "Fuel Efficiency: " << fixed << setprecision(2) << car.calculateFuelEfficiency()
        << " miles/gallon" << endl;
    }
    else if (type == 2) { // ElectricCar
        double batteryCapacity, efficiency;
        cout << "Enter Battery Capacity (kWh) and Efficiency (miles per kWh): ";
        cin >> batteryCapacity >> efficiency;
        ElectricCar ecar(brand, model, mileage, batteryCapacity, efficiency);
        ecar.displayInfo();
        cout << "Range: " << fixed << setprecision(2) << ecar.calculateRange() << " miles";
    }
    else {
        cout << "Invalid vehicle type!" << endl;
    }
    return 0;
}

```

Output:

```

Vehicle Type (1 for Car, 2 for Electric Car): 1
Enter Brand, Model, and Mileage: toyota
corolla
30000
Enter Fuel (gallons) and Distance (miles): 15
300
Vehicle: toyota corolla
Mileage: 30000
Fuel Efficiency: 20.00 miles/gallon

```

Ques 10:

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 <iomanip>

using namespace std;

class Shape {
public:
    virtual float calculateArea() = 0; // Pure virtual function
    virtual ~Shape() {}
};

class Rectangle : public Shape {
private:
    float length, width;
public:
    Rectangle(float l, float w) : length(l), width(w) {}

    float calculateArea() override {
        return length * width;
    }
};

class Circle : public Shape {
private:
    float radius;
public:
    Circle(float r) : radius(r) {}

    float calculateArea() override {
        return 3.14159 * radius * radius;
    }
};
```

```

    }
};

class Triangle : public Shape {
private:
    float base, height;
public:
    Triangle(float b, float h) : base(b), height(h) {}

    float calculateArea() override {
        return 0.5 * base * height;
    }
};

int main() {
    int shapeType;
    float param1, param2;
    cout << "Enter Shape Type (1 for Rectangle, 2 for Circle, 3 for Triangle): ";
    cin >> shapeType;
    Shape* shape = nullptr;
    if(shapeType == 1) {
        // For Rectangle, take length and width
        cout << "Enter Length and Width: ";
        cin >> param1 >> param2;
        shape = new Rectangle(param1, param2);
        cout << "Shape: Rectangle\n";
    }
    else if(shapeType == 2) {
        // For Circle, take radius
        cout << "Enter Radius: ";
        cin >> param1;
        shape = new Circle(param1);
    }
}

```

```

        cout << "Shape: Circle\n";
    }
    else if(shapeType == 3) {
        cout << "Enter Base and Height: ";
        cin >> param1 >> param2;
        shape = new Triangle(param1, param2);
        cout << "Shape: Triangle\n";
    }
    cout << fixed << setprecision(2) << "Area: " << shape->calculateArea() << endl;
    delete shape;
    return 0;
}

```

Output:

Output

Enter Shape Type (1 for Rectangle, 2 for Circle, 3 for Triangle): 1
Enter Length and Width: 5 10
Shape: Rectangle
Area: 50.00

=== Code Execution Successful ===