

## DOMAIN WINTER WINNING CAMP

Name : Jobanjeet Singh

Uid : 22BCS15377

Section : 620- B

Date: 19-DEC-2024

### VERY EASY

#### 1) Sum of Natural Numbers up to N

**Code:**

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

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

    if (n > 0) {
        int sum = n * (n + 1) / 2; // Using the formula
        cout << "The sum of natural numbers from 1 to " << n << " is: " << sum <<
        endl;
    } else {
        cout << "Please enter a positive integer!" << endl;
    }

    return 0;
}
```

## Output

```
Enter a positive integer: 4  
The sum of natural numbers from 1 to 4 is: 10
```

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

## 2. Check if a Number is Prime

```
#include <iostream>
using namespace std;
int main() {
    int num, i;
    bool isPrime = true;
    cout << "Enter a number: ";
    cin >> num;
    if (num <= 1) {
        isPrime = false;
    } else {
        for (i = 2; i <= num / 2; i++) {
            if (num % i == 0) {
                isPrime = false;
                break;
            }
        }
    }
    if (isPrime) {
        cout << "The number is prime." << endl;
    } else {
        cout << "The number is not prime." << endl;
    }
    return 0;
}
```

```
Enter a number: 34
The number is not prime.
```

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

## EASY

### 3. Count Digits in a Number

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

```
int main() {
    int num, count = 0;
    cout << "Enter a positive integer: ";
    cin >> num;
```

```
    while (num != 0) {
        num /= 10;
        count++;
    }
```

```
    cout << "The total number of digits is: " << count << endl;
    return 0;
```

```
}
```

```
Enter a positive integer: 232422322422422
The total number of digits is: 10
```

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

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".

```
#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" << endl;
    else
        cout << "Not a Palindrome Number" << endl;
    return 0;
}
```

#### Output

```
Enter a number: 11
Palindrome Number
```

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

## 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;

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

double area(double length, double width) {
    return length * width;
}

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

int main() {
    double radius, length, width, base, height;
    cout << "Enter the radius of the circle: ";
    cin >> radius;
    cout << "Area of the circle: " << area(radius) << endl;
    cout << "Enter the length and width of the rectangle: ";
    cin >> length >> width;
    cout << "Area of the rectangle: " << area(length, width) << endl;
    cout << "Enter the base and height of the triangle: ";
    cin >> base >> height;
    cout << "Area of the triangle: " << area(base, height, true) << endl;
    return 0;
}
```

```
Output
Enter the radius of the circle: 2
Area of the circle: 12.5664
Enter the length and width of the rectangle: 12
2
Area of the rectangle: 24
Enter the base and height of the triangle: 21
2
Area of the triangle: 21
|

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

### Ques 6:

**Function Overloading for finding maximum of two numbers, three numbers and two floating number.**

### **Code**

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

int max(int a, int b) {
    return (a > b) ? a : b;
}

int max(int a, int b, int c) {
    return (a > b) ? ((a > c) ? a : c) : ((b > c) ? b : c);
}

float max(float a, float b) {
    return (a > b) ? a : b;
}

int main() {
    int choice;
    cout << "Choose the operation:\n";
    cout << "1. Maximum of two integers\n";
```

```

cout << "2. Maximum of three integers\n";
cout << "3. Maximum of two floating-point numbers\n";
cin >> choice;

if (choice == 1) {
    int a, b;
    cout << "Enter two integers: ";
    cin >> a >> b;
    cout << "Maximum of " << a << " and " << b << " is: " << max(a, b) <<
endl;
} else if (choice == 2) {
    int a, b, c;
    cout << "Enter three integers: ";
    cin >> a >> b >> c;
    cout << "Maximum of " << a << ", " << b << " and " << c << " is: " <<
max(a, b, c) << endl;
} else if (choice == 3) {
    float x, y;
    cout << "Enter two floating-point numbers: ";
    cin >> x >> y;
    cout << "Maximum of " << x << " and " << y << " is: " << max(x, y) <<
endl;
} else {
    cout << "Invalid choice!" << endl;
}

return 0;
}

```

## Output

```
Choose the operation:
1. Maximum of two integers
2. Maximum of three integers
3. Maximum of two floating-point numbers
2
Enter three integers: 2 4 23
Maximum of 2, 4 and 23 is: 23

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

## 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
32
2 22
2
23|
Area of Circle: 78.53975
Area of Rectangle: 64.00000
Area of Triangle: 23.00000

=== Code Execution Successful ===

```

**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 =  $\frac{1}{2} \times \text{Base} \times \text{Height}$ ).

```
#include <iostream>

#include <iomanip>

using namespace std;

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

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;  
    if (type == 1) {  
        int l, b;  
        cin >> l >> b;  
        shape = new Rectangle(l, b);  
    } else if (type == 2) {  
        int r;  
        cin >> r;  
        shape = new Circle(r);  
    } else if (type == 3) {
```

```

    int b, h;

    cin >> b >> h;

    shape = new Triangle(b, h);
} else {

    cout << "Invalid shape type";

    return 0;

}

cout << fixed << setprecision(2);

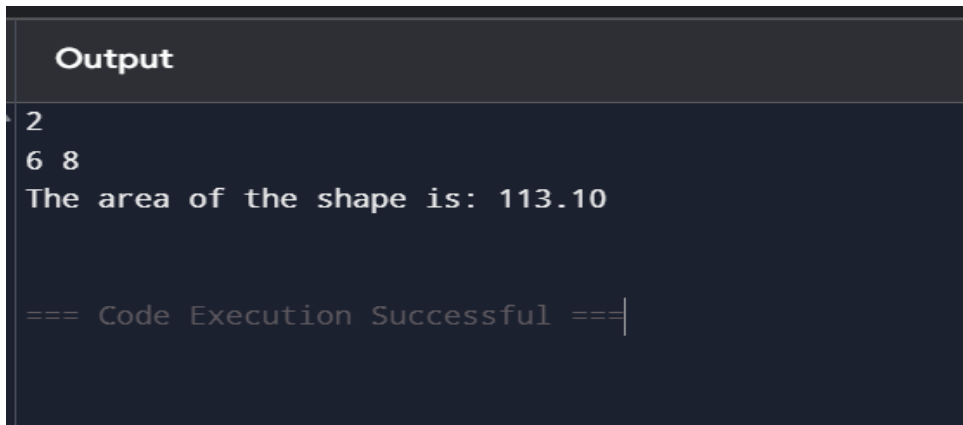
cout << "The area of the shape is: " << shape->getArea() << endl;

delete shape;

return 0;

}

```



The screenshot shows a dark-themed window titled "Output". The output text is as follows:

```

2
6 8
The area of the shape is: 113.10

=== Code Execution Successful ===

```

**VERY HARD**

**Ques 9.**

**Write a C++ program to perform matrix addition and matrix multiplication using function overloading. Implement overloaded**

**functions to handle both operations. The program should prompt the user to enter the dimensions and elements of two matrices and then choose the operation type (1 for addition, 2 for multiplication). The program should then display the result of the chosen operation.**

**Code :**

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

vector<vector<int>> operate(const vector<vector<int>>& A, const
vector<vector<int>>& B, int operationType) {
    int m = A.size(), n = A[0].size();
    vector<vector<int>> result(m, vector<int>(n, 0));
    if (operationType == 1) {
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
                result[i][j] = A[i][j] + B[i][j];
            }
        }
    }
    return result;
}

vector<vector<int>> operate(const vector<vector<int>>& A, const
vector<vector<int>>& B) {
    int m = A.size(), n = A[0].size(), p = B[0].size();
    vector<vector<int>> result(m, vector<int>(p, 0));
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < p; j++) {
            for (int k = 0; k < n; k++) {
                result[i][j] += A[i][k] * B[k][j];
            }
        }
    }
    return result;
}

int main() {
```

```
int m, n, p, operationType;
```

```
cout << "Enter dimensions of Matrix A (m n): ";
```

```
cin >> m >> n;
```

```
vector<vector<int>> A(m, vector<int>(n));
```

```
cout << "Enter elements of Matrix A:\n";
```

```
for (int i = 0; i < m; i++) {
```

```
    for (int j = 0; j < n; j++) {
```

```
        cin >> A[i][j];
```

```
    }
```

```
}
```

```
cout << "Enter dimensions of Matrix B (n p): ";
```

```
cin >> n >> p;
```

```
vector<vector<int>> B(n, vector<int>(p));
```

```
cout << "Enter elements of Matrix B:\n";
```

```
for (int i = 0; i < n; i++) {
```

```
    for (int j = 0; j < p; j++) {
```

```
        cin >> B[i][j];
```

```
    }
```

```
}
```

```
cout << "Choose operation type (1 for Addition, 2 for Multiplication): ";
```

```
cin >> operationType;
```

```
if (operationType == 1) {
```

```
    if (A.size() == B.size() && A[0].size() == B[0].size()) {
```

```
        vector<vector<int>> result = operate(A, B, 1);
```

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

```
        for (const auto& row : result) {
```

```
            for (int elem : row) {
```

```
                cout << elem << " ";
```

```
            }
```

```
            cout << endl;
```

```
        }
```

```
    } else {
```

```
        cout << "Invalid dimensions for operation." << endl;
```

```
    }
```

```

} else if (operationType == 2) {
    if (A[0].size() == B.size()) {
        vector<vector<int>> result = operate(A, B);
        cout << "Result of Matrix Multiplication:\n";
        for (const auto& row : result) {
            for (int elem : row) {
                cout << elem << " ";
            }
            cout << endl;
        }
    } else {
        cout << "Invalid dimensions for operation." << endl;
    }
} else {
    cout << "Invalid operation type." << endl;
}

return 0;}

```

### Ques 10

**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.**

```

#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) {
        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) {

```



```

        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" << endl;
    } else {
        cout << "Invalid vehicle type!" << endl;
    }

    return 0;
}

```

### Output

```

Vehicle Type (1 for Car, 2 for Electric Car): 1
Enter Brand, Model, and Mileage: tata
24 20
Enter Fuel (gallons) and Distance (miles): 18
400
Vehicle: tata 24
Mileage: 20
Fuel Efficiency: 22.22 miles/gallon

```

```

=== Code Execution Successful ===

```