

Name – Daksh
UID – 22BCS15372
Section – 620-B

DAY 1

VERY EASY

1) Print Multiplication Table of a Number

Objective:

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:
 $3 \times 1 = 3, 3 \times 2 = 6, \dots, 3 \times 10 = 30$.

Code:

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

int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= 10; i++) {
        cout << n << " x " << i << " = " << n * i << endl;
    }
    return 0;
}
```

Output:

```
3
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30

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

2)) Print Odd Numbers up to N

Objective:

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.

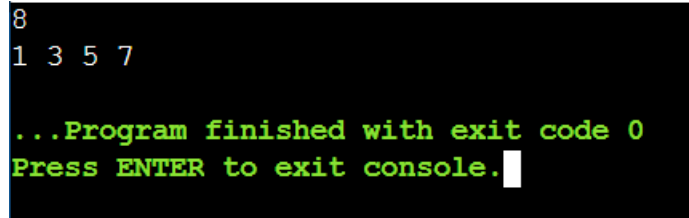
This problem is a simple introduction to loops and conditional checks. The goal is to use a loop to iterate over the numbers and check if they are odd using the condition $i \% 2 \neq 0$.

Code:

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

int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i += 2) {
        cout << i << " ";
    }
    return 0;
}
```

Output:



```
8
1 3 5 7

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

EASY:

3) Count Digits in a Number

Objective:

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>
using namespace std;
```

```
int main() {
    int n, count = 0;
    cin >> n;
    while (n != 0) {
        n /= 10;
        count++;
    }
    cout << count;
    return 0;
}
```

Output:

```
651489
6

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

4) Find the Sum of Digits of a Number

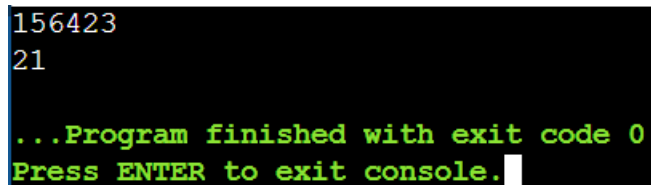
Objective:

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:

```
#include <iostream>
using namespace std;
int main() {
    int n, sum = 0;
    cin >> n;
    while (n != 0) {
        sum += n % 10;
        n /= 10;
    }
    cout << sum;
    return 0;
}
```

Output:



```
156423
21
...Program finished with exit code 0
Press ENTER to exit console.
```

MEDIUM:

5) Function Overloading for Calculating Area.

Objective:

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>
#include <cmath>
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 radius of circle: ";
    cin >> radius;
    cout << "Area of circle: " << area(radius) << endl;

    cout << "Enter length and width of rectangle: ";
    cin >> length >> width;
    cout << "Area of rectangle: " << area(length, width) << endl;

    cout << "Enter base and height of triangle: ";
    cin >> base >> height;
    cout << "Area of triangle: " << area(base, height, true) << endl;

    return 0;
}
```

Output:

```
Enter radius of circle: 5
Area of circle: 78.5397
Enter length and width of rectangle: 4
6
Area of rectangle: 24
Enter base and height of triangle: 3
5
Area of triangle: 7.5

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

6) Polymorphism with Shape Area Calculation.

Objective:

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 <cmath>
using namespace std;

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

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

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

```

    Rectangle(double l, double w) : length(l), width(w) {}
    double area() override {
        return length * width;
    }
};

class Triangle : public Shape {
    double base, height;
public:
    Triangle(double b, double h) : base(b), height(h) {}
    double area() override {
        return 0.5 * base * height;
    }
};

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

    cout << "Enter radius of circle: ";
    cin >> radius;
    Shape* circle = new Circle(radius);
    cout << "Area of circle: " << circle->area() << endl;
    delete circle;

    cout << "Enter length and width of rectangle: ";
    cin >> length >> width;
    Shape* rectangle = new Rectangle(length, width);
    cout << "Area of rectangle: " << rectangle->area() << endl;
    delete rectangle;

    cout << "Enter base and height of triangle: ";
    cin >> base >> height;
    Shape* triangle = new Triangle(base, height);
    cout << "Area of triangle: " << triangle->area() << endl;
    delete triangle;

    return 0;
}

```

Output:

```
Enter radius of circle: 5
Area of circle: 78.5397
Enter length and width of rectangle: 4
6
Area of rectangle: 24
Enter base and height of triangle: 3
5
Area of triangle: 7.5

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

7) Inheritance with Student and Result Classes.

Objective:

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:

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

class Student {
protected:
    int rollNumber;
    string name;

public:
    void getDetails() {
        cout << "Enter roll number: ";
        cin >> rollNumber;
        cin.ignore();
        cout << "Enter name: ";
        getline(cin, name);
    }

    void displayDetails() {
        cout << "Roll Number: " << rollNumber << endl;
```



```

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

class Result : public Student {
    int marks1, marks2, marks3;
    int total;
    float percentage;

public:
    void getMarks() {
        cout << "Enter marks for three subjects: ";
        cin >> marks1 >> marks2 >> marks3;
    }

    void calculateResult() {
        total = marks1 + marks2 + marks3;
        percentage = (total / 3.0);
    }

    void displayResult() {
        displayDetails();
        cout << "Total Marks: " << total << endl;
        cout << "Percentage: " << percentage << "%" << endl;
    }
};

int main() {
    Result studentResult;
    studentResult.getDetails();
    studentResult.getMarks();
    studentResult.calculateResult();
    studentResult.displayResult();

    return 0;
}

```

Output:

```
Enter roll number: 15372
Enter name: Daksh
Enter marks for three subjects: 95 98 97
Roll Number: 15372
Name: Daksh
Total Marks: 290
Percentage: 96.6667%

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

8) Encapsulation with Employee Details

Objective:

Write a program that demonstrates encapsulation by creating a class Employee. The class should have private attributes to store:

Employee ID.

Employee Name.

Employee Salary.

Provide public methods to set and get these attributes, and a method to display all details of the employee.

Code:

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

class Employee {
private:
    int employeeID;
    string employeeName;
    double employeeSalary;

public:
    // Setter methods
    void setEmployeeID(int id) {
        employeeID = id;
    }

    void setEmployeeName(string name) {
        employeeName = name;
    }
}
```

```

void setEmployeeSalary(double salary) {
    employeeSalary = salary;
}

// Getter methods
int getEmployeeID() {
    return employeeID;
}

string getEmployeeName() {
    return employeeName;
}

double getEmployeeSalary() {
    return employeeSalary;
}

// Method to display employee details
void displayDetails() {
    cout << "Employee ID: " << employeeID << endl;
    cout << "Employee Name: " << employeeName << endl;
    cout << "Employee Salary: $" << employeeSalary << endl;
}
};

int main() {
    Employee emp;

    int id;
    string name;
    double salary;

    cout << "Enter Employee ID: ";
    cin >> id;
    emp.setEmployeeID(id);

    cout << "Enter Employee Name: ";
    cin.ignore(); // To clear the buffer
    getline(cin, name);
    emp.setEmployeeName(name);

    cout << "Enter Employee Salary: ";
    cin >> salary;
    emp.setEmployeeSalary(salary);

    cout << "\nEmployee Details:" << endl;

```

```
    emp.displayDetails();  
  
    return 0;  
}
```

Output:

```
Enter Employee ID: 100  
Enter Employee Name: Daksh  
Enter Employee Salary: 100000  
  
Employee Details:  
Employee ID: 100  
Employee Name: Daksh  
Employee Salary: Rs.100000  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

HARD:

9) Implementing Polymorphism for Shape Hierarchies.

Objective:

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 <cmath>
using namespace std;

class Complex {
private:
    double real, imag;

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

    // Function to display complex number in a + bi format
    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& c) {
        return Complex(real + c.real, imag + c.imag);
    }

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

    // Function to calculate magnitude of the complex number
```

```
double magnitude() {  
    return sqrt(real * real + imag * imag);  
}  
};
```

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

```
    cout << "Select the operation:\n";  
    cout << "1. Addition\n";  
    cout << "2. Multiplication\n";  
    cout << "3. Magnitude\n";  
    cout << "Enter your choice: ";  
    cin >> choice;
```

```
    Complex c1, c2;
```

```
    switch (choice) {  
        case 1:  
            cout << "Enter real and imaginary parts of first complex number: ";  
            cin >> r1 >> i1;  
            c1 = Complex(r1, i1);  
            cout << "Enter real and imaginary parts of second complex number: ";  
            cin >> r2 >> i2;  
            c2 = Complex(r2, i2);
```

```
            Complex sum = c1 + c2;  
            cout << "Sum: ";  
            sum.display();  
            break;
```

```
        case 2:  
            cout << "Enter real and imaginary parts of first complex number: ";  
            cin >> r1 >> i1;  
            c1 = Complex(r1, i1);  
            cout << "Enter real and imaginary parts of second complex number: ";  
            cin >> r2 >> i2;  
            c2 = Complex(r2, i2);
```

```
            Complex product = c1 * c2;  
            cout << "Product: ";
```

```

        product.display();
        break;

    case 3:
        cout << "Enter real and imaginary parts of the complex number: ";
        cin >> r1 >> i1;
        c1 = Complex(r1, i1);
        cout << "Magnitude: " << c1.magnitude() << endl;
        break;

    default:
        cout << "Invalid choice\n";
        break;
}

return 0;
}

```

10) Function Overloading for Complex Number Operations.

Objective:

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:

```

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

class Complex {
private:
    double real, imag;

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

```

```

// Function to display complex number in a + bi format
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& c) {
    return Complex(real + c.real, imag + c.imag);
}

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

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

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

    cout << "Select the operation:\n";
    cout << "1. Addition\n";
    cout << "2. Multiplication\n";
    cout << "3. Magnitude\n";
    cout << "Enter your choice: ";
    cin >> choice;

    Complex c1, c2;

    switch (choice) {
        case 1:
            cout << "Enter real and imaginary parts of first complex number: ";

```



```
cin >> r1 >> i1;
c1 = Complex(r1, i1);
cout << "Enter real and imaginary parts of second complex number: ";
cin >> r2 >> i2;
c2 = Complex(r2, i2);

Complex sum = c1 + c2;
cout << "Sum: ";
sum.display();
break;
```

case 2:

```
cout << "Enter real and imaginary parts of first complex number: ";
cin >> r1 >> i1;
c1 = Complex(r1, i1);
cout << "Enter real and imaginary parts of second complex number: ";
cin >> r2 >> i2;
c2 = Complex(r2, i2);

Complex product = c1 * c2;
cout << "Product: ";
product.display();
break;
```

case 3:

```
cout << "Enter real and imaginary parts of the complex number: ";
cin >> r1 >> i1;
c1 = Complex(r1, i1);
cout << "Magnitude: " << c1.magnitude() << endl;
break;
```

default:

```
cout << "Invalid choice\n";
break;
```

```
}
```

```
return 0;
```

```
}
```