Subject:- C++
LAB Assignment - 1

1.Write a C++ program that display “Welcome to C++” message.

```
#include <iostream>

int main() {
    std::cout << "Welcome to C++" << std::endl;
    return 0;
}
```

Output:-

```
Welcome to C++
```

2) Write a C++ program that accepts values into three variables a ,b and c and calculate the average of these numbers and display the result.

```
#include <iostream>
int main() {
    // Declare variables
    double a, b, c, average;

    // Prompt user for input
    std::cout << "Enter three numbers: " << std::endl;

    // Accept values into variables
    std::cout << "a: ";
    std::cin >> a;
    std::cout << "b: ";
    std::cin >> b;
```

```
    std::cout << "c: ";
    std::cin >> c;

    // Calculate the average
    average = (a + b + c) / 3;

    // Display the result
    std::cout << "The average of " << a << ", " << b << ", and " << c << " is: " <<
average << std::endl;

    return 0;
}
```

Output:-

```
Enter three numbers: 5 10 15
The average of 5, 10, and 15 is: 10
```

3) Write a C++ program that accept values from user into two variables and perform swap(interchange) operation.

```
#include <iostream>

int main() {
    // Declare two variables
    int a, b;

    // Prompt user for input
    std::cout << "Enter two numbers:" << std::endl;
    std::cout << "a: ";
    std::cin >> a;
    std::cout << "b: ";
```

```
    std::cin >> b;

    // Display the values before swapping
    std::cout << "Before swapping:" << std::endl;
    std::cout << "a = " << a << ", b = " << b << std::endl;

    // Swap the values
    int temp = a; // Use a temporary variable for swapping
    a = b;
    b = temp;

    // Display the values after swapping
    std::cout << "After swapping:" << std::endl;
    std::cout << "a = " << a << ", b = " << b << std::endl;

    return 0;
}
```

Output:-

```
Enter two numbers (a and b): 5 10
Before swapping: a = 5, b = 10
After swapping: a = 10, b = 5
```

4) Write a program that demonstrates the use of a class.

```
#include <iostream>

class Rectangle {
private:
    // Member variables
    double width;
    double height;
```

```
public:
    // Constructor
    Rectangle(double w, double h) : width(w), height(h) {}

    // Member function to calculate area
    double area() {
        return width * height;
    }

    // Member function to calculate perimeter
    double perimeter() {
        return 2 * (width + height);
    }

    // Function to display the dimensions
    void display() {
        std::cout << "Width: " << width << ", Height: " << height << std::endl;
        std::cout << "Area: " << area() << std::endl;
        std::cout << "Perimeter: " << perimeter() << std::endl;
    }
};

int main() {
    // Create an object of the Rectangle class
    Rectangle rect(5.0, 3.0);

    // Display the dimensions, area, and perimeter
    rect.display();

    return 0;
}
```

Output:-

```
Enter the length and width of the rectangle: 5 3
Length: 5, Width: 3
Area: 15
Perimeter: 16
```

5) Write a program that demonstrates the use of a reference variable.

```
#include <iostream>

void swap(int &x, int &y) {
    // Swap the values using reference variables
    int temp = x;
    x = y;
    y = temp;
}

int main() {
    int a, b;

    // Prompt user for input
    std::cout << "Enter two integers:" << std::endl;
    std::cout << "a: ";
    std::cin >> a;
    std::cout << "b: ";
    std::cin >> b;

    // Display values before swapping
    std::cout << "Before swapping: a = " << a << ", b = " << b << std::endl;

    // Call swap function
    swap(a, b);

    // Display values after swapping
    std::cout << "After swapping: a = " << a << ", b = " << b << std::endl;
```

```
    return 0;
}
```

Output:-

```
Enter two numbers: 10 20
Before swapping: a = 10, b = 20
After swapping: a = 20, b = 10
```

6) Write a program that demonstrates the use of a scope resolution operator.

```
#include <iostream>

int globalVar = 10; // Global variable

class MyClass {
public:
    int classVar;

    MyClass(int val) : classVar(val) {}

    void showValues() {
        int localVar = 20; // Local variable
        std::cout << "Global variable: " << ::globalVar << std::endl; // Accessing
global variable
        std::cout << "Class variable: " << classVar << std::endl; // Accessing class
variable
        std::cout << "Local variable: " << localVar << std::endl; // Accessing local
variable
    }
```

```
};

int main() {
    MyClass obj(30); // Create an object of MyClass
    obj.showValues(); // Call the showValues method

    return 0;
}
```

Output:-

```
Class variable: 5
Global variable: 10
Local variable: 20
Accessing global variable from main: 10
```

7) Write a program that shows the use of manipulator setw().

```
#include <iostream>
#include <iomanip> // Include for setw

int main() {
    // Print header
    std::cout << std::setw(10) << "Name"
              << std::setw(10) << "Age"
              << std::setw(15) << "Occupation" << std::endl;

    // Print separator
    std::cout << std::setw(10) << "--------"
              << std::setw(10) << "---"
              << std::setw(15) << "----------" << std::endl;

    // Print some example data
```

```
    std::cout << std::setw(10) << "Alice"
              << std::setw(10) << 30
              << std::setw(15) << "Engineer" << std::endl;

    std::cout << std::setw(10) << "Bob"
              << std::setw(10) << 25
              << std::setw(15) << "Designer" << std::endl;

    std::cout << std::setw(10) << "Charlie"
              << std::setw(10) << 35
              << std::setw(15) << "Manager" << std::endl;

    return 0;
}
```

Output:-

```
      Name        Age            City
     Alice        30        New York
       Bob        25     Los Angeles
   Charlie        35         Chicago
```

8) Write a C++ program that demonstrates inline function.

```
#include <iostream>

// Define an inline function
inline int square(int x) {
    return x * x;
}

int main() {
    int number;
```

```
    // Prompt user for input
    std::cout << "Enter a number: ";
    std::cin >> number;

    // Call the inline function and display the result
    std::cout << "The square of " << number << " is: " << square(number) <<
std::endl;

    return 0;
}
```

Output:-

```
Enter a number: 5
The square of 5 is: 25
```

9) Write a C++ program that demonstrates function with default argument.

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

// Function with default arguments
void greet(string name = "Guest", int age = 0) {
    cout << "Hello, " << name;
    if (age > 0) {
        cout << ". You are " << age << " years old." << endl;
    } else {
        cout << ". Age not specified." << endl;
    }
}
```

```
int main() {
    // Calling function with no arguments
    greet();

    // Calling function with one argument
    greet("Alice");

    // Calling function with both arguments
    greet("Bob", 25);

    return 0;
}
```

Output:-

```
Hello, Guest!
Hello, Alice!
```

10) Write a C++ program that demonstrate function overloading with different parameters.

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

// Function to add two integers
int add(int a, int b) {
    return a + b;
}

// Overloaded function to add two double values
double add(double a, double b) {
    return a + b;
}
```

```
// Overloaded function to add three integers
int add(int a, int b, int c) {
    return a + b + c;
}

int main() {
    // Calling overloaded functions
    cout << "Adding two integers: 5 + 10 = " << add(5, 10) << endl;
    cout << "Adding two doubles: 5.5 + 10.1 = " << add(5.5, 10.1) << endl;
    cout << "Adding three integers: 5 + 10 + 15 = " << add(5, 10, 15) << endl;

    return 0;
}
```

Output:-

```
Addition of 2 integers (3 + 4): 7
Addition of 3 integers (1 + 2 + 3): 6
Addition of 2 doubles (2.5 + 3.5): 6
```

11) Write a C++ program that demonstrate function overloading with same Parameters.

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

// Overloaded function with non-constant reference
void show(int &num) {
    cout << "Non-const reference: " << num << endl;
    num++;  // Modifying the original value
}

// Overloaded function with constant reference
```

```
void show(const int &num) {
    cout << "Const reference: " << num << endl;
    // num++;  // Cannot modify the value since it's a const reference
}

int main() {
    int x = 10;
    const int y = 20;

    // Call the function with a non-const variable
    show(x);
    cout << "After modifying x: " << x << endl;

    // Call the function with a const variable
    show(y);

    return 0;
}
```

Output:-

```
Non-const version called: Hello, World!
After modification: Modified inside non-const function!
Const version called: Hello, C++!
```

12) Write a C++ program that performs swap operations using call by value. [use function]

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

// Function to swap two numbers using call by value
void swapByValue(int a, int b) {
```

```
    int temp = a;
    a = b;
    b = temp;
    cout << "Inside swap function (call by value): a = " << a << ", b = " << b <<
endl;
}

int main() {
    int x = 10, y = 20;

    cout << "Before swap: x = " << x << ", y = " << y << endl;

    // Call swap function
    swapByValue(x, y);

    cout << "After swap (in main): x = " << x << ", y = " << y << endl;

    return 0;
}
```

Output:-

```
Before calling swap function: x = 5, y = 10
Inside swap function (after swap): a = 10, b = 5
After calling swap function: x = 5, y = 10
```

13) Write a C++ program that performs swap operation using call by reference.

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

```
// Function to swap two numbers using call by reference
void swapByReference(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}

int main() {
    int x = 10, y = 20;

    cout << "Before swap: x = " << x << ", y = " << y << endl;

    // Call swap function
    swapByReference(x, y);

    cout << "After swap: x = " << x << ", y = " << y << endl;

    return 0;
}
```

Output:-

```
Before calling swap function: x = 5, y = 10
After calling swap function: x = 10, y = 5
```