Q. 1. Write CPP Program to find maximum of 3 numbers using inline function.

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

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

int main() {
  int a, b, c;
  cin >> a >> b >> c;
  cout << "The maximum of " << a << ", " << b << ", and " << c << " is " << max(a, b, c) << endl;
  return 0;
}
```

Q. 2. Write a C++ program to create a class Item with data members Item_Code, Item_Name, Item_Price. Write member functions to accept and display Item information also display number of objects created for a class. (Use Static data member and Static member function)

```
#include<iostream.h>
#include<conio.h>
class item
{
int code;
char inm[20];
float price;
static int cnt;
public:
void accept();
void show();
static void count();
};
int item::cnt;
void item::accept()
```

```
{
cout<<"\n Enter Item code : ";
cin>>code;
cout<<"\n Enter Item Name : ";
cin>>inm;

cout<<"\n Enter Item price : ";
cin>>price;

cnt++;
}
void item::show()
{
cout<<"\n Code : "<<code;
 cout<<"\n Name : "<<inm;
 cout<<"\n Price : "<<price;
 }
void item::count()
{ cout<<"\n Total Count : "<<cnt;
 }
void main()
{
int i,n;
clrscr();
item a[20];
cout<<"\n How Many item you want : ";
cin>>n;
for(i=0;i<n;i++)
 {
a[i].accept();
cout<<"\n-------------------------------------";
}
for(i=0;i<n;i++)
 {
a[i].show();
cout<<"\n-------------------------------------";
}
item::count();
getch();
}
</n;i++)
</n;i++)
</cnt;
</price;
```

</inm;
</code;

Q 3. Write a C++ program to swap two integer values and two float values by using function template.

```
#include <iostream>

using namespace std;


template <typename T>

void swap(T& a, T& b) {

  T temp = a;

  a = b;

  b = temp;

}


int main() {

  int x = 10, y = 20;

  float a = 3.14, b = 2.71;


  cout << "Before swapping:" << endl;

  cout << "x = " << x << ", y = " << y << endl;

  cout << "a = " << a << ", b = " << b << endl;


  swap(x, y);

  swap(a, b);


  cout << "\nAfter swapping:" << endl;

  cout << "x = " << x << ", y = " << y << endl;

  cout << "a = " << a << ", b = " << b << endl;
```

```
  return 0;
}
```

Output

Before swapping:

x = 10, y = 20

a = 3.14, b = 2.71

After swapping:

x = 20, y = 10

a = 2.71, b = 3.14

Q. 4. Create a class distance which accept the distances in feet & inches from the user & display the sum of two distances in feet & inch. [use object as function argument, Write the function inside the class.

```
#include <iostream>

class Distance {
private:
    int feet;
    int inches;

public:
    // Constructor to initialize the object
    Distance(int ft = 0, int in = 0) : feet(ft), inches(in) {}

    // Member function to add distances
    Distance addDistances(const Distance& otherDistance) const {
        Distance result;
```

```
        // Calculate the sum of feet and inches
        result.feet = feet + otherDistance.feet;
        result.inches = inches + otherDistance.inches;

        // Adjust inches if it exceeds 12
        if (result.inches >= 12) {
            result.feet += result.inches / 12;
            result.inches %= 12;
        }

        return result;
    }

    // Member function to display the distance
    void displayDistance() const {
        std::cout << "Distance: " << feet << " feet " << inches << " inches" << std::endl;
    }
};

int main() {
    // Get distances from the user
    int feet1, inches1, feet2, inches2;

    std::cout << "Enter the feet for the first distance: ";
    std::cin >> feet1;
    std::cout << "Enter the inches for the first distance: ";
    std::cin >> inches1;
```

```
    std::cout << "Enter the feet for the second distance: ";
    std::cin >> feet2;
    std::cout << "Enter the inches for the second distance: ";
    std::cin >> inches2;

    // Create Distance objects
    Distance distance1(feet1, inches1);
    Distance distance2(feet2, inches2);

    // Add the distances and display the result
    Distance resultDistance = distance1.addDistances(distance2);
    resultDistance.displayDistance();

    return 0;
}
```

Q. 5. Write a C++ program to overload function volume and find volume of cube, cylinder and sphere.

```
#include<iostream>
using namespace std;
#define PI 3.1416

// function prototypes
float volume(float length, float breadth, float height);
float volume(float radius);
float volume(float radius, float height);

int main(){
        float cube_l = 40.0, cube_b = 30.0, cube_h = 10.0;
        float sphere_r = 2.5;
```

```
        float cylinder_r = 2.5, cylinder_h = 10.0;
        cout<<"Volume of Cube ="<<volume(cube_l, cube_b, cube_h)<<endl;
        cout<<"Volume of Sphere ="<<volume(sphere_r)<<endl;
        cout<<"Volume of Cylinder ="<<volume(cylinder_r, cylinder_h)<<endl;
        return 0;
}
// function defination
float volume(float length, float breadth, float height){
        return length * breadth * height;
}
float volume(float radius){
        return (4.0/3.0) * PI * radius * radius *radius;
}
float volume(float radius, float height){
        return PI * radius *radius * height;
}
```

output

Sample Run:

Volume of Cube =12000

Volume of Sphere =65.45

Volume of Cylinder =196.35

Q. 6. Write a C++ program create a calculator for an arithmetic operator (+, -, *, /). The program should take two operands from user and performs the operation on those two operands depending upon the operator entered by user. Use a switch statement to select the operation.

```
/*Write a C++ program create a calculator for an arithmetic operator
(+, -, *, /). The program should take two operands from user and
performs the operation on those two operands depending upon the
operator entered by user. Use a switch statement to select the
operation. Finally, display the result. Some sample interaction with
the program might look like this:
```

```
Enter first number, operator, second number: 10 / 3
Answer = 3.333333
Do another (y/n)? y
Enter first number, operator, second number: 12 + 100
Answer = 112
Do another (y/n)? n
*/

#include<iostream>
using namespace std;
class Calculator
{
   private:
          float num1,num2,result;
          char op;
   public:
          void get();
          void calculate();
};
void Calculator::get()
{
   cout<<"\nEnter first number, operator, second number:\n";
   cin>>num1;
   cin>>op;
   cin>>num2;
}
void Calculator::calculate()
{

   switch(op)
   {
      case '+':
               result=num1+num2;
                  cout<<" Answer = "<<result;

               break;
      case '-':
                    result=num1-num2;
                  cout<<" Answer = "<<result;

               break;
      case '*':
                result=num1*num2;
                  cout<<" Answer = "<<result;

                break;
      case '/':
               if(num2==0)
               cout<<"\n Error. Not valid.";
                result=num1/num2;
                  cout<<" Answer = "<<result;

      break;
   }

}
int main()
```

```
{
   char ag;
   Calculator obj;
   x:obj.get();
   obj.calculate();
   cout<<"\n Do another (y/n)? ";
   cin>>ag;
   if(ag=='y'||ag=='Y')
   goto x;
   return 0;
}

/*OUTPUT:
student@student-OptiPlex-3010:~$ g++ groupa5.cpp
student@student-OptiPlex-3010:~$ ./a.out

Enter first number, operator, second number:
10/3
 Answer = 3.33333
 Do another (y/n)? y

Enter first number, operator, second number:
12+100
 Answer = 112
 Do another (y/n)? n*/
```

Q. 7. Write a C++ program to count and display the number of lines not starting with alphabet 'A' present in a text file "STORY.TXT".

Example: If the file "STORY.TXT" contains the following lines,

“The roses are red. A girl is playing there. There is a playground. An aeroplane is in the sky.

```
Write a function in C++ to count and display the number of lines not
starting with
alphabet 'A' present in a text file "STORY.TXT".
Example:
If the file "STORY.TXT" contains the following lines,
The roses are red.
A girl is playing there.
There is a playground.
An aeroplane is in the sky.
Numbers are not allowed in the password.
The function should display the output as 3.


#include<iostream>
#include<fstream>
using namespace std;
int main()
{
    ifstream fin;
```

```
    fin.open("Story.txt");

    char str[100];
    int count=0;
    int count1=0;

    while(!fin.eof())
    {
        fin.getline(str,100);
        if(str[0]!='A')
        {
            count++;
        }
        else if(str[0]=='A'&&str[1]==' ')
        {
            count1++;
        }
    }

    cout<<"The number of lines not starting with 'A'
are:"<<count<<"\n";
    cout<<"The number of lines starting with 'A' are:"<<count1<<"\n";

    fin.close();
    return 0;
}
```

Q. 8. Write a C++ program to calculate area of cone, sphere and circle by using function overloading

```
#include <iostream>

#include <cmath>


const double PI = 3.141592653589793;


class AreaCalculator {

public:

  // Function to calculate the area of a circle

  double calculateArea(double radius) {

    return PI * radius * radius;

  }


  // Function to calculate the area of a sphere

  double calculateArea(double radius, char shape) {
```

```
        if (shape == 's') {
            // Area of a sphere
            return 4 * PI * radius * radius;
        } else if (shape == 'c') {
            // Area of a cone (assuming height is 1 for simplicity)
            return PI * radius * (radius + sqrt(radius * radius + 1));
        } else {
            // Invalid shape
            std::cout << "Invalid shape code. Please use 's' for sphere or 'c' for cone." << 
std::endl;
            return 0.0;
        }
    }
};

int main() {
    AreaCalculator calculator;

    // Calculate and display the area of a circle
    double circleRadius;
    std::cout << "Enter the radius of the circle: ";
    std::cin >> circleRadius;
    std::cout << "Area of the circle: " << calculator.calculateArea(circleRadius) << std::endl;

    // Calculate and display the area of a sphere
    double sphereRadius;
    std::cout << "Enter the radius of the sphere: ";
    std::cin >> sphereRadius;
    std::cout << "Area of the sphere: " << calculator.calculateArea(sphereRadius, 's') << 
std::endl;
```

```
    // Calculate and display the area of a cone
    double coneRadius;
    std::cout << "Enter the radius of the cone: ";
    std::cin >> coneRadius;
    std::cout << "Area of the cone: " << calculator.calculateArea(coneRadius, 'c') << std::endl;

    return 0;
}
```

Q. 9. Create a C++ class for a student object with the following attributes—roll no, name, number of subjects, marks of subjects. Write member function for accepting marks and display all information of student along with total and Percentage. Display mark list with Use of manipulators.

```
#include <iostream>
#include <iomanip> // for using manipulators

class Student {
private:
    int rollNo;
    std::string name;
    int numSubjects;
    int* marks;

public:
    // Constructor to initialize the object
    Student(int roll, const std::string& studentName, int subjects)
        : rollNo(roll), name(studentName), numSubjects(subjects) {
        marks = new int[numSubjects];
    }
```

```
    // Destructor to free dynamic memory
    ~Student() {
        delete[] marks;
    }

    // Member function to accept marks for each subject
    void acceptMarks() {
        std::cout << "Enter marks for each subject:" << std::endl;
        for (int i = 0; i < numSubjects; ++i) {
            std::cout << "Subject " << i + 1 << ": ";
            std::cin >> marks[i];
        }
    }

    // Member function to display student information, total, and percentage
    void displayInfo() const {
        std::cout << "Roll No: " << rollNo << std::endl;
        std::cout << "Name: " << name << std::endl;

        // Display mark list using manipulators
        std::cout << std::setw(15) << std::left << "Subject" << std::setw(10) << std::right << "Marks" << std::endl;
        for (int i = 0; i < numSubjects; ++i) {
            std::cout << std::setw(15) << std::left << "Subject " + std::to_string(i + 1) << std::setw(10) << std::right << marks[i] << std::endl;
        }

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

```
            total += marks[i];
        }

        double percentage = static_cast<double>(total) / numSubjects;

        std::cout << "Total Marks: " << total << std::endl;
        std::cout << "Percentage: " << std::fixed << std::setprecision(2) << percentage << "%" << std::endl;
    }
};

int main() {
    // Get student information from the user
    int rollNo, numSubjects;
    std::string name;

    std::cout << "Enter Roll No: ";
    std::cin >> rollNo;

    std::cout << "Enter Name: ";
    std::cin.ignore(); // Ignore the newline character in the buffer
    std::getline(std::cin, name);

    std::cout << "Enter the number of subjects: ";
    std::cin >> numSubjects;

    // Create a Student object
    Student student(rollNo, name, numSubjects);

    // Accept marks and display information
```

```
    student.acceptMarks();
    student.displayInfo();

    return 0;
}
```

Q. 10. Write a C++ program to make a simple calculator using class template.

```
#include <iostream>

template <typename T>
class Calculator {
private:
    T num1;
    T num2;

public:
    Calculator(T a, T b) : num1(a), num2(b) {}

    T add() const {
        return num1 + num2;
    }

    T subtract() const {
        return num1 - num2;
    }

    T multiply() const {
        return num1 * num2;
    }
```

```
    T divide() const {
        if (num2 != 0) {
            return num1 / num2;
        } else {
            std::cerr << "Error: Division by zero." << std::endl;
            return static_cast<T>(0);
        }
    }
};

int main() {
    // Example usage of the Calculator class template with integer values
    Calculator<int> intCalculator(10, 5);

    std::cout << "Addition: " << intCalculator.add() << std::endl;
    std::cout << "Subtraction: " << intCalculator.subtract() << std::endl;
    std::cout << "Multiplication: " << intCalculator.multiply() << std::endl;
    std::cout << "Division: " << intCalculator.divide() << std::endl;

    // Example usage of the Calculator class template with double values
    Calculator<double> doubleCalculator(15.5, 3.5);

    std::cout << "\nAddition: " << doubleCalculator.add() << std::endl;
    std::cout << "Subtraction: " << doubleCalculator.subtract() << std::endl;
    std::cout << "Multiplication: " << doubleCalculator.multiply() << std::endl;
    std::cout << "Division: " << doubleCalculator.divide() << std::endl;

    return 0;
```

}

Q. 11. Write a C++ program to accept length and width of a rectangle. Calculate and display perimeter as well as area of a rectangle by using Inline function.

```
#include <iostream>

// Inline function to calculate the area of a rectangle
inline double calculateArea(double length, double width) {
    return length * width;
}

// Inline function to calculate the perimeter of a rectangle
inline double calculatePerimeter(double length, double width) {
    return 2 * (length + width);
}

int main() {
    double length, width;

    // Accept length and width from the user
    std::cout << "Enter the length of the rectangle: ";
    std::cin >> length;

    std::cout << "Enter the width of the rectangle: ";
    std::cin >> width;

    // Calculate and display the area and perimeter using inline functions
    std::cout << "Area of the rectangle: " << calculateArea(length, width) << std::endl;
    std::cout << "Perimeter of the rectangle: " << calculatePerimeter(length, width) <<
std::endl;
```

```
    return 0;
}
```

Q. 12. Write a C++ program to concatenate two Strings using operator + function.

```
#include <iostream>
#include <string>

class Concatenator {
private:
    std::string str1;
    std::string str2;

public:
    Concatenator(const std::string& s1, const std::string& s2) : str1(s1), str2(s2) {}

    // Overloading + operator to concatenate strings
    std::string operator+() const {
        return str1 + str2;
    }
};

int main() {
    std::string string1, string2;

    // Input two strings from the user
    std::cout << "Enter the first string: ";
    std::getline(std::cin, string1);
```

```
    std::cout << "Enter the second string: ";
    std::getline(std::cin, string2);

    // Create an object of Concatenator
    Concatenator concatenator(string1, string2);

    // Concatenate and display the result
    std::string result = +concatenator;
    std::cout << "Concatenated String: " << result << std::endl;

    return 0;
}
```

Q. 13. Write a program to derive a class rectangle from base class shape using single inheritance.

```
#include <iostream>

// Base class
class Shape {
public:
    virtual void displayArea() const = 0; // Pure virtual function for area calculation
};

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

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

```
    // Implementation of the pure virtual function to calculate the area of a rectangle
    void displayArea() const override {
        double area = length * width;
        std::cout << "Area of the rectangle: " << area << std::endl;
    }
};

int main() {
    // Create an object of the derived class Rectangle
    Rectangle rectangle(5.0, 8.0);

    // Call the displayArea function of the derived class
    rectangle.displayArea();

    return 0;
}
```

Q. 14. Write a C++ program to swap two float values by using function template.

```
#include <iostream>

// Function template to swap two values
template <typename T>
void swapValues(T& a, T& b) {
    T temp = a;
    a = b;
    b = temp;
}

int main() {
    float value1, value2;
```

```
    // Input two float values from the user
    std::cout << "Enter the first float value: ";
    std::cin >> value1;

    std::cout << "Enter the second float value: ";
    std::cin >> value2;

    // Display the values before swapping
    std::cout << "\nBefore swapping:" << std::endl;
    std::cout << "Value 1: " << value1 << std::endl;
    std::cout << "Value 2: " << value2 << std::endl;

    // Swap the values using the function template
    swapValues(value1, value2);

    // Display the values after swapping
    std::cout << "\nAfter swapping:" << std::endl;
    std::cout << "Value 1: " << value1 << std::endl;
    std::cout << "Value 2: " << value2 << std::endl;

    return 0;
}
```

Q. 15. Write a C++ program to overload function volume and find volume of cube, cylinder and sphere.

```
#include <iostream>
#include <cmath>

const double PI = 3.141592653589793;

// Function overload for volume of a cube
double volume(double side) {
   return std::pow(side, 3);
}
```

```
// Function overload for volume of a cylinder
double volume(double radius, double height) {
    return PI * std::pow(radius, 2) * height;
}

// Function overload for volume of a sphere
double volume(double radius) {
    return (4.0 / 3.0) * PI * std::pow(radius, 3);
}

int main() {
    double side, radius, height;

    // Input for cube
    std::cout << "Enter the side length of the cube: ";
    std::cin >> side;
    std::cout << "Volume of the cube: " << volume(side) << std::endl;

    // Input for cylinder
    std::cout << "Enter the radius of the cylinder: ";
    std::cin >> radius;
    std::cout << "Enter the height of the cylinder: ";
    std::cin >> height;
    std::cout << "Volume of the cylinder: " << volume(radius, height) << std::endl;

    // Input for sphere
    std::cout << "Enter the radius of the sphere: ";
    std::cin >> radius;
    std::cout << "Volume of the sphere: " << volume(radius) << std::endl;
```

```
    return 0;
}
```

Q. 16. Write a CPP program to copy the contents of one file to another.

```
#include <iostream>
#include <fstream>

int main() {
    std::ifstream inputFile("input.txt");  // Input file
    std::ofstream outputFile("output.txt");  // Output file

    // Check if the input file is open
    if (!inputFile.is_open()) {
        std::cerr << "Error: Unable to open the input file." << std::endl;
        return 1;
    }

    // Check if the output file is open
    if (!outputFile.is_open()) {
        std::cerr << "Error: Unable to open the output file." << std::endl;
        return 1;
    }

    char ch;

    // Copy contents from input file to output file
    while (inputFile.get(ch)) {
        outputFile.put(ch);
```

```
    }

    // Close the files
    inputFile.close();
    outputFile.close();

    std::cout << "File contents copied successfully." << std::endl;

    return 0;
}
```

Q. 17 Define a class Animal with their basic features as class members. Create two derived classes from Animal named herbivores and Carnivores (type) with their own features too. Accept name of animal with type and display all the related information.

```
#include <iostream>
#include <string>

// Base class Animal
class Animal {
protected:
    std::string name;
    std::string type;

public:
    // Constructor to initialize name and type
    Animal(const std::string& animalName, const std::string& animalType)
        : name(animalName), type(animalType) {}

    // Member function to display basic information
    void displayInfo() const {
        std::cout << "Name: " << name << std::endl;
        std::cout << "Type: " << type << std::endl;
```

```
    }
};

// Derived class Herbivore
class Herbivore : public Animal {
private:
    std::string herbivoreFeature;

public:
    // Constructor to initialize herbivoreFeature
    Herbivore(const std::string& animalName, const std::string& feature)
        : Animal(animalName, "Herbivore"), herbivoreFeature(feature) {}

    // Member function to display herbivore-specific information
    void displayHerbivoreInfo() const {
        displayInfo();
        std::cout << "Herbivore Feature: " << herbivoreFeature << std::endl;
    }
};

// Derived class Carnivore
class Carnivore : public Animal {
private:
    std::string carnivoreFeature;

public:
    // Constructor to initialize carnivoreFeature
    Carnivore(const std::string& animalName, const std::string& feature)
        : Animal(animalName, "Carnivore"), carnivoreFeature(feature) {}

    // Member function to display carnivore-specific information
```

```
    void displayCarnivoreInfo() const {
        displayInfo();
        std::cout << "Carnivore Feature: " << carnivoreFeature << std::endl;
    }
};

int main() {
    // Example usage
    Herbivore herbivore("Elephant", "Large Ears");
    Carnivore carnivore("Lion", "Sharp Claws");

    // Display information for herbivore
    std::cout << "Herbivore Information:" << std::endl;
    herbivore.displayHerbivoreInfo();
    std::cout << std::endl;

    // Display information for carnivore
    std::cout << "Carnivore Information:" << std::endl;
    carnivore.displayCarnivoreInfo();

    return 0;
}
```

Q 18. Write a C++ program to find area and perimeter of rectangle using Inline function.

```
#include <iostream>

class Rectangle {
private:
    double length;
    double width;
```

```
public:
    // Constructor to initialize the rectangle with length and width
    Rectangle(double len, double wid) : length(len), width(wid) {}

    // Inline function to calculate the area of the rectangle
    inline double calculateArea() const {
        return length * width;
    }

    // Inline function to calculate the perimeter of the rectangle
    inline double calculatePerimeter() const {
        return 2 * (length + width);
    }
};

int main() {
    double length, width;

    // Input length and width from the user
    std::cout << "Enter the length of the rectangle: ";
    std::cin >> length;

    std::cout << "Enter the width of the rectangle: ";
    std::cin >> width;

    // Create a Rectangle object
    Rectangle rectangle(length, width);

    // Calculate and display the area and perimeter using inline functions
```

```
    std::cout << "Area of the rectangle: " << rectangle.calculateArea() << std::endl;
    std::cout << "Perimeter of the rectangle: " << rectangle.calculatePerimeter() << std::endl;

    return 0;
}
```

Q. 19. Write a C++ program to find area and volume of cylinder using Inline function.

```
#include <iostream>
#include <cmath>

class Cylinder {
private:
    double radius;
    double height;

public:
    // Constructor to initialize the cylinder with radius and height
    Cylinder(double r, double h) : radius(r), height(h) {}

    // Inline function to calculate the surface area of the cylinder
    inline double calculateArea() const {
        return 2 * M_PI * radius * (radius + height);
    }

    // Inline function to calculate the volume of the cylinder
    inline double calculateVolume() const {
        return M_PI * std::pow(radius, 2) * height;
    }
```

```
};

int main() {
    double radius, height;

    // Input radius and height from the user
    std::cout << "Enter the radius of the cylinder: ";
    std::cin >> radius;

    std::cout << "Enter the height of the cylinder: ";
    std::cin >> height;

    // Create a Cylinder object
    Cylinder cylinder(radius, height);

    // Calculate and display the area and volume using inline functions
    std::cout << "Surface Area of the cylinder: " << cylinder.calculateArea() << std::endl;
    std::cout << "Volume of the cylinder: " << cylinder.calculateVolume() << std::endl;

    return 0;
}
```

Q. 20. Write a C++ program for Exception Handling Divide by zero Using C++ Programming.

```
#include <iostream>
#include <stdexcept>

int main() {
    try {
        int numerator, denominator;
```

```
        // Input numerator and denominator from the user
        std::cout << "Enter the numerator: ";
        std::cin >> numerator;

        std::cout << "Enter the denominator: ";
        std::cin >> denominator;

        // Check if the denominator is zero
        if (denominator == 0) {
            throw std::runtime_error("Error: Division by zero is not allowed.");
        }

        // Perform the division and display the result
        double result = static_cast<double>(numerator) / denominator;
        std::cout << "Result of the division: " << result << std::endl;
    } catch (const std::exception& e) {
        // Catch and handle exceptions
        std::cerr << e.what() << std::endl;
    }

    return 0;
}
```

Q. 21. Write a C++ program to swap two integer values by using function template.

```
#include <iostream>

// Function template to swap two values
template <typename T>
void swapValues(T& a, T& b) {
    T temp = a;
```

```
    a = b;
    b = temp;
}

int main() {
    int value1, value2;

    // Input two integer values from the user
    std::cout << "Enter the first integer value: ";
    std::cin >> value1;

    std::cout << "Enter the second integer value: ";
    std::cin >> value2;

    // Display the values before swapping
    std::cout << "\nBefore swapping:" << std::endl;
    std::cout << "Value 1: " << value1 << std::endl;
    std::cout << "Value 2: " << value2 << std::endl;

    // Swap the values using the function template
    swapValues(value1, value2);

    // Display the values after swapping
    std::cout << "\nAfter swapping:" << std::endl;
    std::cout << "Value 1: " << value1 << std::endl;
    std::cout << "Value 2: " << value2 << std::endl;

    return 0;
}
```