**Problem 1: Inline Functions and Preprocessor Directives**

**Task:**  
Write a program that calculates the area of a rectangle using an inline function. Additionally, define a preprocessor macro that prints debugging information when the program is compiled in debug mode.

**Requirements:**

1. Implement an inline function calculateArea(int width, int height) that returns the area.
2. Use a preprocessor directive (#define DEBUG) to conditionally print debugging information when compiling in debug mode.
3. Demonstrate the usage of both the inline function and the preprocessor directive in main().

**Problem 2: Name Mangling Demonstration**

**Task:**  
Write a program that defines two functions with the same name but different parameters in C++. Compile the program and inspect the symbol names using nm or objdump (Linux) or dumpbin (Windows).

**Requirements:**

1. Define two overloaded functions void print(int) and void print(double).
2. Compile the program and inspect the decorated names using nm (objdump) a.out | grep print (or equivalent on Windows - *dumpbin*).
3. Explain the differences in the generated names.

**Problem 3: Comparing Execution Speed of Macro, Inline, and Non-Inline Functions**

**Task:**

Write a program that compares the execution time of three different approaches for calculating the square of a number:

1. **A macro (#define SQUARE(x))**
2. **An inline function (inline int squareInline(int x))**
3. **A regular (non-inline) function (int squareNotInline(int x))**

The program should run each approach in a loop for **10 million iterations**, measure the execution time using std::chrono, and print the results.

**Requirements:**

* Define a macro SQUARE(x) that computes the square of x.
* Implement an **inline function** squareInline(int x).
* Implement a **regular function** squareNotInline(int x).
* Use std::chrono::high\_resolution\_clock to measure execution time for each approach.
* Run each method inside a loop **10 million times**.
* Compare and print the execution times.

**Problem 4: Function Call Logger Using Preprocessor Macros**

**Task:**

Write a program that defines a **preprocessor macro** to:

1. **Automatically call a function** and
2. **Print the name of the function being called** before executing it.

**Requirements:**

* Define a macro CALL\_FUNC(func, args...) that:
  + Prints "Calling function: func" before calling func(args...).
  + Calls the function with its provided arguments.
* Implement at least two different functions (e.g., add(int, int) and multiply(int, int)).
* Use the macro to call both functions in main().

Example output:

*Calling function: add*

*Result: 7*

*Calling function: multiply*

*Result: 12*

**Problem 5: Organizing a Calculator Program with Multiple Files**

**Task:**

Create a simple calculator program that supports addition, subtraction, multiplication, and division, and organize the code into multiple files:

1. **Calculator Operations**
2. **Main Function**
3. **Header File for Declarations**

The program should include a **header file** to declare the functions and use **multiple source files** for the implementation of the calculator operations.

**Requirements:**

1. **Create three files**:
   * **calculator.h**: A header file containing the declarations of functions (add, subtract, multiply, and divide).
   * **calculator.cpp**: A source file where the actual functions are defined.
   * **main.cpp**: The main file where the user can select the operation and input numbers.
2. **Functions to be implemented**:
   * int add(int a, int b) – Returns the sum of a and b.
   * int subtract(int a, int b) – Returns the difference between a and b.
   * int multiply(int a, int b) – Returns the product of a and b.
   * double divide(int a, int b) – Returns the quotient of a divided by b. If division by zero occurs, return a message.
3. **Usage of the header file**:
   * The header file should declare the functions, and the source files should include it.
4. **Implementation details**:
   * The main program should prompt the user to select an operation and input two numbers.
   * Based on the selection, the corresponding function from the calculator.cpp file should be called.
   * Print the result of the operation.