C++ Basics Programming Assignment

Section 1: Basic Syntax and Data Types (15 points)

1. (3 pts) Explain the purpose of the main() function in a C++ program

The starting point for every C++ program. When run the program, the computer loads it into memory and starts executing instructions from main(). At the end, it returns an integer (usually 0) to the operating system to show if the program finished successfully or if there was an error.

2. (4 pts) List four fundamental data types in C++ and briefly describe each

- int: used for storing whole numbers (integers).
- char: stores a single character, like a letter or symbol.
- bool: holds a boolean value, which is just true or false.
- double: used for floating-point numbers (decimals) with double precision.

4. (4 pts) Explain the difference between float and double. Include one situation where double is preferred

The main difference is precision. A float is a single-precision floating-point number, while a double is double-precision. This means double takes up more space but is much more precise.

Use double when you need more accuracy in your calculations. Also, C++ treats decimal numbers like 3.14 as doubles by default, so it's usually the standard choice unless you're trying to save memory.

Section 2: Control Structures (20 points)

6. (5 pts) Explain the difference between a while loop and a do-while loop. Include a short code snippet for each

- **`while` loop:** Checks condition before the loop runs. If false right at the start, the code inside the loop will never run.

```
int i = 0;
while (i < 5) {
    cout << i << " ";
    i++;
}
```

- **`do-while` loop:** Runs the code first, and then checks the condition. This guarantees that the loop runs at least once, even if the condition is false.

```
int i = 0;
do {
    cout << i << " ";
    i++;
} while (i < 5);
```

Section 3: Functions (20 points)

9. (4 pts) Write a function declaration and definition for a function `calculateArea` that takes two double parameters (length and width) and returns the area

```
// Function Declaration
double calculateArea(double length, double width);

// Function Definition
double calculateArea(double length, double width) {
    return length * width;
}
```

10. (4 pts) Explain function overloading in C++. Provide a small example with at least two overloaded functions

When you have multiple same name functions, but different parameters, the compiler tries to choose which one to use based on the arguments you pass it.

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

int area(int side) {
return side \* side;
}

int area(int length, int width) {
return length \* width;
}

int main() {
cout << "Square: " << area(5) << endl;
cout << "Rectangle: " << area(5, 10) << endl;
return 0;
}
```

11. (4 pts) Explain the difference between pass-by-value and pass-by-reference. When should pass-by-reference be used?

Difference:

- Pass-by-value: The function gets a copy of the variable. If you change it inside the function, the original variable outside stays the same. This is the default for things like int.
- Pass-by-reference: The function gets a reference to the actual variable's memory address. If you change it inside the function, it changes the original variable too.

When to use Pass-by-Reference:

- Modifying Data: If you actually want the function to update the variable .
- Efficiency: If you are passing a really big object, passing by reference saves time because the computer doesn't have to copy the whole thing.

12. (4 pts) What are default arguments in C++? Write a function example that uses a default argument

Default arguments are values assigned to function parameters in the function declaration. If the caller omits the argument for that parameter, the default value is used automatically.

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

void displaySum(int a, int b = 10) {
    cout << "Sum: " << a + b << endl;
}

int main() {
    displaySum(5);
    displaySum(5, 20);
    return 0;
}
```

Section 4: Object-Oriented Programming (20 points)

15. (4 pts) Explain encapsulation in C++. How is it implemented in a class?

Explanation: Encapsulation is bundling variables (data) and functions (methods) together into a single unit called a class. It also involves "data hiding," which means protecting the data so it can't be messed up by code outside the class.

Implementation: We usually make the variables private so they can't be accessed directly. Then, we make public functions (getters and setters) to let outside code read or change the data safely.

16. (4 pts) Explain the difference between public, private, and protected access specifiers

- public: Anything declared public can be accessed anywhere: inside the class, inside derived classes, or in main().
- private: These can only be accessed inside the class itself. They are hidden from everything else.
- protected: These are like private members, but with one difference: derived classes (child classes) can access them. Outside code like main() still can't touch them.

17. (6 pts) Explain inheritance in C++. Write a simple example where a class Square inherits from Rectangle

Inheritance is when a new class (Child/Derived class) takes on the properties and functions of an existing class (Parent/Base class). It's really useful for reusing code so you don't have to write the same thing twice.

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

class Rectangle {
protected:
    double length;
    double width;
public:
    void setValues(double l, double w) {
        length = l;
        width = w;
    }
    double getArea() {
        return length * width;
    }
};

class Square : public Rectangle {
public:
    void setSide(double side) {
        setValues(side, side);
    }
};

int main() {
    Square s;
    s.setSide(4.0);

    cout << "Area of Square: " << s.getArea() << endl;

    return 0;
}
```

Section 5: Memory Management and Pointers (15 points)

19. (5 pts) What is a memory leak? Explain two ways to prevent memory leaks in C++

A memory leak happens when allocate memory on the heap (using new) but forget to free it (using delete). The program keeps holding onto that memory even though it's not using it, which can slow things down or make the program crash.

Two ways to prevent it:

- Pair new and delete: Whenever you write new, make sure there's a matching delete to free that memory.
- Use Smart Pointers/Containers: Instead of managing memory manually, use things like std::vector or smart pointers (unique_ptr, shared_ptr). They handle the memory cleanup automatically when the variable goes out of scope.

Section 6: Error Handling and const Usage (10 points)

21. (5 pts) Explain exception handling in C++. Write a simple try-catch example that handles division by zero

Exception handling is a way to catch errors (exceptions) so the program doesn't just crash immediately. It lets you transfer control to a specific block of code to handle the problem.

- try: Put the risky code here.
- throw: Signals that an error happened.
- catch: Catches the error and decides what to do with it (like printing a message).

22. (5 pts) Explain the purpose of the const keyword. Write a short C++ program demonstrating a constant variable and a function parameter declared as const

The const keyword makes a variable constant, meaning its value can't be changed once it's set.