Task 10: Function Templates in C++

Objective:
To understand and implement function templates in C++. This task demonstrates how a single template function can be used to compare two values of different data types and return the larger value.

1. Concept Overview

Function templates allow writing generic functions that work with different data types without rewriting the same logic. The compiler generates type-specific versions of the function during compilation.

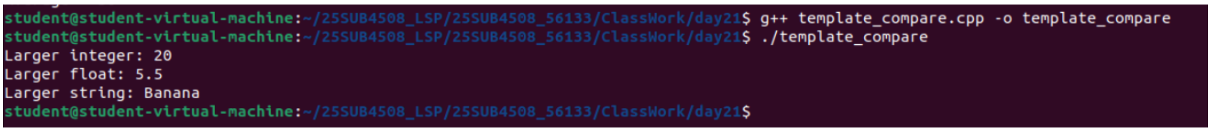
2. Program Code

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

 template <typename T>
 T compare(T a, T b) {
        return (a > b) ? a : b;
 }

 int main() {
        int i1 = 10, i2 = 20;
        float f1 = 5.5, f2 = 2.3;
        string s1 = "Apple", s2 = "Banana";

        cout << "Larger integer: " << compare(i1, i2) << endl;
        cout << "Larger float: " << compare(f1, f2) << endl;
        cout << "Larger string: " << compare(s1, s2) << endl;

        return 0;
 }
```

3. Compilation Instructions

```
g++ template_compare.cpp -o template_compare
```

4. Sample Output



```
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day21$ g++ template_compare.cpp -o template_compare
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day21$ ./template_compare
Larger integer: 20
Larger float: 5.5
Larger string: Banana
student@student-virtual-machine:~/25SUB4508_LSP/25SUB4508_56133/ClassWork/day21$
```

5. Observations & Explanation

1. The compare() function is defined as a template using the template keyword.
2. The same function logic works for int, float, and string data types.
3. The '>' operator is used for comparison; for strings, lexicographical comparison is performed.
4. The compiler generates separate versions of the function for each data type.

6. Advantages of Function Templates

- Code reusability
- Type safety
- Reduced code duplication

7. Conclusion

This task demonstrates the power of function templates in C++. Using templates allows writing generic, reusable, and efficient code that works seamlessly with multiple data types.