Experiment #1

Templates in C++ Programming Language

Student's Name:	
Semester:	Date:

Assessment:

Assessment Point	**Weight**	**Grade**
Methodology and correctness of results		
Discussion of results		
Participation		
Assessment Points' Grade:		

Comments:

Experiment #1:

Templates in C++ Programming Language

Objectives:

1. To introduce the students with the templates
2. To implement function templates
3. To implement class templates
4. To understand the advantages of templates in generic programming

Discussion:

Templates are powerful features of C++ which allows you to write generic programs. In simple terms, you can create a single function or a class to work with different data types using templates.

The concept of templates can be used in two different ways:

- Function Templates
- Class Template

Function Templates

A function template works in a similar to a normal function, with one key difference. A single function template can work with different data types at once but, a single normal function can only work with one set of data types. Normally, if you need to perform identical operations on two or more types of data, you use function overloading to create two functions with the required function declaration. However, a better approach would be to use function templates because you can perform the same task writing less and maintainable code.

Syntax

```
template <class T>

T someFunction(T arg)

{

   ... .. ...

}
```

In the above code, T is a template argument that accepts different data types (int, float), and **class** is a keyword. When, an argument of a data type is passed to someFunction(), compiler generates a new version of someFunction() for the given data type.

Example 1: Swap Data Using Function Templates

```
#include <iostream>
using namespace std;
template <typename T>
void Swap(T &n1, T &n2)
{
        T temp;
        temp = n1;
        n1 = n2;
        n2 = temp;
}
void main()
{
        int i1 = 1, i2 = 2;
        float f1 = 1.1, f2 = 2.2;
        char c1 = 'a', c2 = 'b';
        cout << "Before passing data to function template.\n";
        cout << "i1 = " << i1 << "\ni2 = " << i2;
        cout << "\nf1 = " << f1 << "\nf2 = " << f2;
        cout << "\nc1 = " << c1 << "\nc2 = " << c2;
        Swap(i1, i2);
        Swap(f1, f2);
        Swap(c1, c2);
      cout << "\n\nAfter passing data to function template.\n";
        cout << "i1 = " << i1 << "\ni2 = " << i2;
        cout << "\nf1 = " << f1 << "\nf2 = " << f2;
        cout << "\nc1 = " << c1 << "\nc2 = " << c2;
}
```

Output

```
Before passing data to function template.
i1 = 1
i2 = 2
f1 = 1.1
f2 = 2.2
c1 = a
c2 = b
After passing data to function template.
i1 = 2
i2 = 1
f1 = 2.2
f2 = 1.1
c1 = b
c2 = a
```

Exercise 1:

Write a c++ program to find maximum of two data items using function template?

Pseudo code:

```
T  max(T a,T b)
begin
        if (a>b)
                return a
        else
                return b
        end
```

Output:

```
enter two characters:
a  c
c
enter a,b:
5 7
7
enter p,q:
5.2 7.5
7.5
```

Solution of Exercise 1

Exercise 2:

Write a program to explain class template by creating a template T for a class named pair having two data members of type T which are inputted by a constructor and a member function get-max() return the greatest of two numbers to main. Note: the value of T depends upon the data type specified during object creation.