Lab Questions

Q1. Write a function template swapValues() that swaps two variables of any data type. Demonstrate its use with int, float, and char.

```
#include <iostream>

using namespace std;

template <typename T>

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

    T temp = a;

    a = b;

    b = temp;

}

int main() {

    int x = 5, y = 10;

    float a = 1.5, b = 2.5;

    char c1 = 'A', c2 = 'B';

    swapValues(x, y);

    cout << "Swapped int: " << x << " " << y << endl;

    swapValues(a, b);

    cout << "Swapped float: " << a << " " << b << endl;

    swapValues(c1, c2);

    cout << "Swapped char: " << c1 << " " << c2 << endl;

    return 0;

}
```

```
Swapped int: 10 5
Swapped float: 2.5 1.5
Swapped char: B A

Process returned 0 (0x0)   execution time : 0.107 s
Press any key to continue.
```

Q2. Write a program to overload a function template maxValue() to find the maximum of two values (for same type) and three values (for same type). Call it using int, double, and char.

```
#include <iostream>

using namespace std;

template <typename T>

T maxValue(T a, T b) {

    return (a > b) ? a : b;

}

template <typename T>

T maxValue(T a, T b, T c) {

    return maxValue(maxValue(a, b), c);

}

int main() {

    int x = 10, y = 20, z = 15;

    double p = 3.5, q = 7.2, r = 6.1;

    char c1 = 'a', c2 = 'z', c3 = 'm';

    cout << "Max of 2 ints: " << maxValue(x, y) << endl;

    cout << "Max of 3 ints: " << maxValue(x, y, z) << endl;

    cout << "Max of 2 doubles: " << maxValue(p, q) << endl;

    cout << "Max of 3 doubles: " << maxValue(p, q, r) << endl;

    cout << "Max of 2 chars: " << maxValue(c1, c2) << endl;

    cout << "Max of 3 chars: " << maxValue(c1, c2, c3) << endl;

    return 0;

}
```

```
Max of 2 ints: 20
Max of 3 ints: 20
Max of 2 doubles: 7.2
Max of 3 doubles: 7.2
Max of 2 chars: z
Max of 3 chars: z

Process returned 0 (0x0)   execution time : 0.090 s
Press any key to continue.
```

Q3. Create a class template Calculator<T> that performs addition, subtraction, multiplication, and division of two data members of type T. Instantiate it with int and float.

```
#include <iostream>
using namespace std;
template <typename T>
class Calculator {
    T a, b;
public:
    Calculator(T x, T y) : a(x), b(y) {}
    T add() { return a + b; }
    T subtract() { return a - b; }
    T multiply() { return a * b; }
    T divide() { return a / b; }
};
int main() {
    Calculator<int> intCalc(10, 5);
    cout << "Int Add: " << intCalc.add() << endl;
    cout << "Int Subtract: " << intCalc.subtract() << endl;
    cout << "Int Multiply: " << intCalc.multiply() << endl;
    cout << "Int Divide: " << intCalc.divide() << endl;
    Calculator<float> floatCalc(7.5f, 2.5f);
    cout << "Float Add: " << floatCalc.add() << endl;
    cout << "Float Subtract: " << floatCalc.subtract() << endl;
    cout << "Float Multiply: " << floatCalc.multiply() << endl;
    cout << "Float Divide: " << floatCalc.divide() << endl;
    return 0;
}
```

```
Int Add: 15
Int Subtract: 5
Int Multiply: 50
Int Divide: 2
Float Add: 10
Float Subtract: 5
Float Multiply: 18.75
Float Divide: 3

Process returned 0 (0x0)   execution time : 0.085 s
Press any key to continue.
```

Q4. Define a class template Base<T> with a protected data member and a member function to display it. Derive a class Derived<T> from it, add another data member, and display both data members. Use string and int types to test.

```
#include <iostream>

using namespace std;

template <typename T>

class Base {

protected:

    T baseData;

public:

    Base(T data) : baseData(data) {}

    void displayBase() {

        cout << "Base data: " << baseData << endl;

    }};

template <typename T>

class Derived : public Base<T> {

    T derivedData;

public:

    Derived(T baseVal, T derivedVal) : Base<T>(baseVal), derivedData(derivedVal) {}

    void displayBoth() {

        cout << "Base data: " << this->baseData << endl;

        cout << "Derived data: " << derivedData << endl;

    }};

int main() {

    Derived<int> intObj(100, 200);

    intObj.displayBoth();

    Derived<string> stringObj("BaseString", "DerivedString");

    stringObj.displayBoth();

    return 0;

  }
```

```
Base data: 100
Derived data: 200
Base data: BaseString
Derived data: DerivedString

Process returned 0 (0x0)   execution time : 0.096 s
Press any key to continue.
```

Q5. Write a program to demonstrate the use of Container, Iterator, and Algorithm components in a single program using a vector<int> and performing sorting using sort().

```
#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

vector<int> numbers = {23, 10, 45, 15, 5};

cout << "Original Vector Elements:\n";

vector<int>::iterator it;

for (it = numbers.begin(); it != numbers.end(); ++it) {

cout << *it << " ";

}

sort(numbers.begin(), numbers.end());

cout << "\n\nSorted Vector Elements (Ascending):\n";

for (it = numbers.begin(); it != numbers.end(); ++it) {

cout << *it << " ";

}

return 0;

}
```

```
Original Vector Elements:
23 10 45 15 5

Sorted Vector Elements (Ascending):
5 10 15 23 45
Process returned 0 (0x0)   execution time : 0.100 s
Press any key to continue.
```

6. Write a program to use the STL algorithm functions: sort(), reverse(), find(), and count() on a vector<int>.

```
#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

vector<int> data = {5, 2, 8, 2, 1, 9, 2};

cout << "Original Vector Elements:\n";

for (int val : data)

cout << val << " ";

sort(data.begin(), data.end());

cout << "\n\nSorted Vector (Ascending):\n";

for (int val : data)

cout << val << " ";

reverse(data.begin(), data.end());

cout << "\n\nVector after Reversing:\n";

for (int val : data)

cout << val << " ";

auto it = find(data.begin(), data.end(), 8);

if (it != data.end())

cout << "\n\nElement 8 found at position: " << (it - data.begin());

else

cout << "\n\nElement 8 not found.";

int countTwos = count(data.begin(), data.end(), 2);\

cout << "\n\nNumber of times 2 appears: " << countTwos << endl;

return 0;

}
```

```
Original Vector Elements:
5 2 8 2 1 9 2

Sorted Vector (Ascending):
1 2 2 2 5 8 9

Vector after Reversing:
9 8 5 2 2 2 1

Element 8 found at position: 1

Number of times 2 appears: 3

Process returned 0 (0x0)   execution time : 0.102 s
Press any key to continue.
```

Discussions:

While doing the lab assignments, we found out the significances of the use of templates in C++. We have to be careful about the use of syntax while writing the code. We also got the idea of using different STL algorithm functions like : sort(), reverse(), find(), and count().

Conclusions:

And hence we successfully implemented the use of template functions and STL algorithm functions in C++.