CS3201: OBJECT ORIENTED PROGRAMMING LABORATORY

Topic: Function Templates, Class templates, Unary Operator Overload, Revision

Lab: 06 **Date: 03/05/2024**

OBSERVATION

1. Which of the following is true about template?
   1) Template is a feature of C++ that allows us to write one code for different data types.
   2) We can write one function that can be used for all data types including user defined types. Like sort(), max(), min(), ..etc.
   3) We can write one class or struct that can be used for all data types including user defined types. Like Linked List, Stack, Queue ..etc.
   4) Template is an example of compile time polymorphism.

2. Predict the output:

```
#include <iostream>
using namespace std;
template <typename T>
void fun(const T&x)
{
   static int count = 0;
   cout << \"x = \" << x << \" count = \" << count << endl;
   ++count;
   return;
}
int main()
{
   fun<int> (1);
   cout << endl;
   fun<int>(1);
   cout << endl;
   fun<double>(1.1);
   cout << endl;
   return 0;
}
```

3. What will be the output for the following program?

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

template <class T>
class Test
{
```

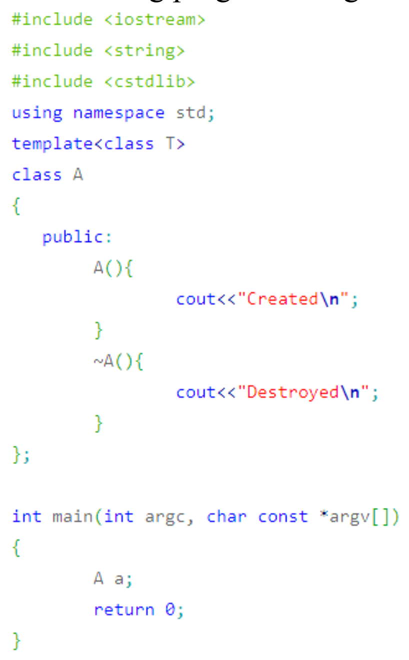
```
private:
    T val;
public:
    static int count;
    Test()  {   count++;   }
};

template<class T>
int Test<T>::count = 0;

int main()
{
    Test<int> a;
    Test<int> b;
    Test<double> c;
    cout << Test<int>::count   << endl;
    cout << Test<double>::count << endl;
    return 0;
}
```

4. Differentiate between function template and class template (tabular form). Why are they used?

5. Debug the following program and give the proper output.



```
#include <iostream>
#include <string>
#include <cstdlib>
using namespace std;
template<class T>
class A
{
   public:
        A(){
                cout<<"Created\n";
        }
        ~A(){
                cout<<"Destroyed\n";
        }
};

int main(int argc, char const *argv[])
{
        A a;
        return 0;
}
```

EXECUTION QUESTIONS

1. Implement a Polynomial class template that supports addition, subtraction, and multiplication of polynomials.

2. Implement a function template mergeSort() to sort an array using the Merge Sort algorithm.

3. Implement a C++ class `TextProcessor` to represent a text string. The class should have private data members `text` of type `string`. Do the following:
   - Overload the unary `!` operator as a friend function to return the number of vowels in the text.
   - Overload the unary `~` operator as a friend function to return the number of consonants in the text.

   Write a C++ program to demonstrate the use of this class by counting the number of vowels and consonants in a given text string. Additionally, overload the unary `+` operator as a friend function to return the text in *uppercase*.

4. Imagine you're developing a data processing application for a financial institution. The application needs to handle various types of financial transactions, including those involving different numeric data types such as integers, floating-point numbers, and currency values.

   Design a function template `calculateTotal()` that can compute the total amount of transactions for any data type. The function should take an array of transaction amounts and the number of transactions as parameters and return the total amount.

   Implement the function template and demonstrate its usage by calculating the total amount of transactions for the following scenarios:

   1. Total amount of transactions for a week represented by an array of integers.

   2. Total amount of transactions for a month represented by an array of floating-point numbers.

   3. **(BONUS QUESTION)** Total amount of transactions for a year represented by an array of currency values (you can represent currency values using a custom `Currency` class or struct).

   Ensure to use the *function template* to handle different data types and validate your implementation with appropriate test cases.

5. Write a C++ program that implements a template function `swap` that takes two generic parameters (variables) and swaps their values. Override the `swap` function to specialize it for integers, allowing the swapping of integer variables to be done more efficiently.