**LALITPUR ENGINEERING COLLEGE**

**DEPARTMENT OF COMPUTER ENGINEERING**

**SUBJECT: OBJECT ORIENTED PROGRAMMING IN C++**

**LABSHEET-6**

**Objective:** To familiarize students with generic programming and templates.

**Theory:**

**Generic programming** is about generalizing software components so that they can be easily reused in a wide variety of situations. In C++, class and function templates are particularly effective mechanisms for generic programming because they make the generalization possible without sacrificing efficiency.

**Templates** are a way of making your classes more abstract by letting you define the behavior of the class without actually knowing what datatype will be handled by the operations of the class.

The basic syntax for declaring a template class is as follows:

**template <class a\_type> class a\_class {...};**

The keyword 'class' above simply means that the identifier a type will stand for a datatype. NB: a type is not a keyword; it is an identifier that during the execution of the program will represent a single datatype.

For example, you could, when defining variables in the class, use the following line:

**a\_type a\_var;**

and when the programmer defines which datatype 'a\_type' is to be when the program instantiates a particular instance of a class, a var will be of that type.

When defining a function as a member of a template class, it is necessary to define it as a template function:

**template<class a\_type> void a\_class<a\_type>::a\_function(){...}**

When declaring an instance of a template class, the syntax is as follows:

**a\_class<int> an\_example\_class;**

**Exception handling:** An exception is a problem that arises during the execution of a program. A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide by zero.Exceptions provide a way to transfer control from one part of a program to another. C++ exception handling is built upon three keywords: **try, catch,** and **throw**.

* **throw:** A program throws an exception when a problem shows up. This is done using a **throw** keyword.
* **catch:** A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The **catch** keyword indicates the catching of an exception.
* **try:** A **try** block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.

Assuming a block will raise an exception, a method catches an exception using a combination of the **try** and **catch** keywords. A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch looks like the following:

try

{

// protected code

}catch( Exception Name e1 )

{

// catch block

}catch( ExceptionName e2 )

{

// catch block

}catch( ExceptionName eN )

{

// catch block

}

**QUESTIONS**

1. Write a program to create a vector class for finding scalar product of two generic data type (int and float)

2. Write a program to swap any two generic data type using function template

3. Write a program to input two numbers and divide the first number by their difference, also check exception for divide by 0.