**Project 2 -** CSC 402/502

OO vs Templates

We are going to write a comparison of the two major programming paradigms we have learned in C++: OO and templates (generic programming). Simply, you will generate two vectors of 100,000 random values each and then sort them. The vectors will store objects that wrap other types, in this case ints and strings. You will need to write some code to handle the generation and sorting.

Object-Oriented Version

Create one cpp file that contains an OO-oriented way of comparing numbers. Use an interface called IComparable that declares a function isLessThan that will compare this object to another of the same interface type. Next write two classes that implement the interface. These will be simple "wrapper classes" to store ints and strings, respectively. This is similar to how Integer is an object wrapper for the primitive int. Write whatever methods are required inside them; follow good OO-design principles and C++ syntax.

Next, you will need two non-class functions, one for sorting and one for sorting verification. Each will accept a vector of the type of your interface and make use of the interface's isLessThan to determine ordering. Keep in mind that during grading I may include a class that I write that implements IComparable. Make sure your code is not hard-coded to use your classes in any way.

Write a main to test your program. It must create two vectors of 100,000 objects each, one storing objects of your int wrapper class, the other of your string wrapper class. The ints should be values 0-1000, which you should already know how to generate. What about strings? I'm not putting any constraints on them other than their content must be random and must be 1 to 4 characters long. Next sort and verify each using your functions. The last piece of the program is that the sorting of the two vectors should be timed. Output said time to the screen at the end of the program.

Visit this link to learn how to time a function call:

http://www.cplusplus.com/reference/ctime/clock/

Template Version

Create one cpp file that provides a generic programming alternative to the above OO program. Re-use your implementation classes here, but alter them so they no longer use the interface. Adjust your sorting and verification functions so they are written as templates and make any adjustments to main that may be required. Your two functions will no longer use isLessThan though, they will assume operator< is defined. For extra credit, write your implementation to use functors instead. Make sure you time the sorting in this program as well. Graduate students MUST use functors instead.

Results

After your code is implemented (correctly), run each of the programs with 100 items in the list, then 1,000, then 10,000, then 100,000. For each size, run each version 3 times in an effort to eliminate random fluctuations. Try to do all the runs back-to-back to eliminate environment inconsistencies. Track the times each takes, either in a spreadsheet (preferred) or a well formatted text file.

Can you come to any conclusion based on this data?

Is one implementation faster than the other?

What if you were to time how long the program takes to run as a whole, rather than just the sorting portion? This is not required, but if you want to actually test this, you can run in linux and make use of the *time* command: time myprogram.exe or time java –jar myprogram.jar.

Answer these questions, along with any other relevant thoughts you have, in a simple summary document (this can be a simple text file).

#### Submission

Place all of your source code files inside a zip along with your timing results and summary document(s) and submit on Blackboard.