Lab 11: Templates  
In this lab you will:

* Learn about C++ templates
* Write a simple templated class

Basic Structure  
C++ provides class templates in order to create generic classes that can work with any data type.

In this lab, you will implement a Pair class template. This is similar to what was discussed in lecture (although that was a template with two type parameters). Try to first do this lab without referring back to the lecture notes. You will learn more that way.

A Pair is a data structure that supports holding two objects of any specified type. It supports a standard default constructor and the expected 2-argument constructor, as well as accessors for the two data members, called first() and second(). It will also support an overloaded operator==() that compares two Pair objects in an order-independent fashion; in other words, they are equal if a.first == b.first and a.second == b.second **or** a.first == b.second and a.second == b.first.

Getting the Files  
If you've ssh’ed into the GL servers, issue the following command at the command prompt while you’re inside your lab11 directory:

cp /afs/umbc.edu/users/c/m/cmarron/pub/cmsc202/lab11.cpp .

(Note the '.' at the end of the command--that is very important.)

You will have the following file:

1. [lab11.cpp:](http://www.csee.umbc.edu/courses/undergraduate/202/fall15_marron/labs/lab11/lab11.cpp) contains the driver program.

The Pair class  
The first step is to write a non-templated Pair class. The class must have the following methods:

* Pair(): a default constructor that does nothing.
* Pair(int first, int second): a 2-argument constructor that creates a pair with the given values
* first(): an accessor that returns the value of the first member of the pair.
* second(): an accessor that returns the value of the second member of the pair.
* operator==(): overload the equality operator to compare two pairs in a position-independent fashion (i.e., the first and second can match in any order).

It will also need private data members to hold the two values.

As a reminder, here is the syntax for declaring the overloaded "==" operator:

bool operator== (const Pair& rhs) const;

and the syntax for the function definition:

bool Pair::operator== (const Pair& rhs) const {

/\* Code here to see if first==first and second==second,

\* OR first==second and second==first

\*/

}

Once you have completed the non-templated version of the Pair class, compile it with lab11.cpp to test it:

g++ -Wall -ansi lab11.cpp Pair.h -o lab11

The Templated Pair class  
Once you have successfully tested the simple Pair class, you must modify it to be a templated class. This is mainly a matter of adding template <class T> at the appropriate locations and changing types from int to T. In particular, the implementation of the overloaded "==" operator changes to:

bool Pair<T>::operator== (const Pair<T>& rhs) const {

/\* Code here to see if first==first and second==second,

\* OR first==second and second==first

\*/

}

You will also need to change the structure of Pair.h and Pair.cpp:

* Pair.cpp should be guarded and include Pair.h at the *top* of the file.
* Pair.h should be guarded and include Pair.cpp at the *bottom* of the file.

To test the templated class, comment out the declarations of pair1, pair2, pair3, and pair4 on lines 8 – 11, *uncomment* the declarations on lines 13 – 16, and compile:

g++ -Wall -ansi lab11.cpp Pair.cpp -o lab11