Data Structures (810:052) Lab 3 Name:_____

Objectives: You will gain experience:

- using and implementing C++ exception handling
- implementing C++ template functions

Download the following file to your desktop: http://www.cs.uni.edu/~fienup/cs052s10/labs/lab3.zip
Extract this file by right-clicking on lab3.zip icon and selecting Extract All.

Part A: In lecture 4 we discussed the `try-catch` mechanism to aid in programming for error conditions. We looked at a Rectangle (Program 16.5 in the text) class with exceptions for negative width and length. For Part A of this lab, I want you to modify the `IntArray` class similarly to handle a "subscript error" by throwing an exception with an exception class object defined with the IntArray class. The `lab3.zip` file you downloaded and extracted contains a `IntArray` folder (and a Rectangle fold) with a Visual Studio C++ project file: `IntArray.sln` inside. Double-click on it to open this project in Visual Studio. Run the current main program (in `main.cpp`).

You'll need to modify the `IntArray`class by:

- Adding a SubscriptError exception class definition inside the `IntArray`class of `IntArray.h`
- Remove the subscriptError function in`IntArray.cpp`. Replace all calls to the subscriptError function by throwing the SubscriptError exception.
- Modify the main program in `main.cpp` to utilize the `try-catch` four times -- once for each IntArray usage of its operator[]. In each `catch` part print a meaningful error message.

After you have debugged your program, raise your hand and demonstrate your program.

Part B: A *function template* is a "generic" function that can work with any data type. The programmer writes the function with a parameter for a data type, e.g., "<class T>" below. The compiler generates the function(s) with the needed type(s). For example, below is the template function for selection sort.

```
template <class T>
void selectionSort(T array[], int size) {
      T temp;
      int testIndex, minIndex, firstUnsortedIndex;

      for (firstUnsortedIndex = 0; firstUnsortedIndex < size-1; firstUnsortedIndex++) {
            minIndex = firstUnsortedIndex;
            for (testIndex = firstUnsortedIndex+1; testIndex < size; testIndex++) {
                  if (array[testIndex] < array[minIndex]) {
                        minIndex = testIndex;
                  } // end if
            } // end for
            temp = array[firstUnsortedIndex];
            array[firstUnsortedIndex] = array[minIndex];
            array[minIndex] = temp;
      } // end for
} // end selectionSort
```

Notice that only the first three lines are different than the "int" selection sort from class. Your tasks is to convert the bubbleSort function into a template function. Code for be found in the lab3/BubbleSort folder Modify the main program to call bubble sort twice: with an array of "int"s and with an array of "double"s.

After you have debugged your program, raise your hand and demonstrate your program.

If you complete all parts of the lab, nothing needs to be turned in for this lab. If you do not get done today, then show me the completed lab in next week's lab period. Make sure that you log off the computer before you leave.