CS273 Module 3 Project

Make the sorting race – and throw in the kitchen sink!

# Objectives:

By the end of this project you should:

1. Orchestrate most if not all you have learned since CS171…
2. In concert with functors and the STL (Standard Template Library), particularly vectors and linked lists…
3. To write an application that allows a human user to select which sorting algorithms to race against each other!

# Due dates:

11/1 by end of day

# Instructions:

By now you have written three sorts, shown some skill with inheritance and polymorphism, designed using ADTs and associated tools, and done some unit testing. Believe it or not, with the tools you’ll be initiated to this module you’ll be able to incorporate pretty much everything you’ve learned to date since CS171 to write a substantial application. Don’t worry about the scope of the project! If you trust your **team** (yes, this project you can work in teams if you want to!) to the tools we’ve been talking about since the semester’s beginning, you’ll be able to do this and feel very proud of yourself once you’re done.

**Step 1:** Read these instructions carefully. Come talk to Matt if anything feels confusing. Do this earlier rather than later.

**Step 2:** Study up on functors, vectors, linked lists, and (if you need to learn this) how to query the system time to find out how long your algorithm takes. (Matt will post tutorials on all this.)

**Step 3:** “Translate” your sorting algorithms from M1P so that they 1) sort vectors and 2) are implemented as **functors**. (As you do this, you’ll want to lean heavily into the software design and development tools we’ve been talking about – writing use cases, using UML, designing pseudocode, and writing unit tests as well as implementing your sorts.)

**Step 4:** Design an application that allows a user to select from among the sorting algorithms you’ve just translated into functors. Your application should set up a linked list of the functors selected by the user, then use an iterator to walk through that list invoking each sort one at a time to sort the numbers stored in random\_number.txt (see Blackboard). Your algorithm should measure how long each sort took and display that time to the screen in a nifty format.

**Additional requirements:**

* Each sort, upon commencing, should sort a fresh vector. You don’t want your first algorithm to do all the work and the others then to cheat by “sorting” something that’s already been sorted!
* Your opening menu must have the following options:
  + Add a sorting algorithm to the list to be run.
  + Display which sorts have been added to the list by name.
  + Run all sorts (run the sorting race) and display the times for each algorithm.
* When your sorting race is being executed, you should display to the screen which algorithm ran and how long it ran for each algorithm selected by the user.

**Some help and hints!**

I’ve written some seed code with polymorphic class hierarchy including only Bubble Sort + an example of how to make a linked list of sorting functors + how to run them. You may use this code as much as you want BUT you must 1) acknowledge, per the department’s academic honesty policy, your use; 2) extend and modify it to fully realize the requirements of this module project; 3) in the process also produce the requisite use cases, UML, pseudocode, and test cases that highlights and documents your team’s accomplishment. I’ll make this seed code available to you on Github here:

https://github.com/MTBellAtWhitworth/SortFunctors

# Deliverables

Documentation including use cases, UML, pseudocode, and test cases. Also, a completed implementation that may or may not make cited re-use of any code given to you Matt.

Good luck, and may the functors be with you!