Lab 3: Searching & STL Containers

CSCI 41

Objectives:

- Practice timing functions and differentiating computational complexities
- Get experience with the C++ standard template library containers that we'll be using/implementing in this class

Part 1: Get the starter code

First, `cd` into the folder where you want to store your class files (e.g., `~/labs`). Copy the starter code there with the following line:

```
gimme lab03@csci41
```

This will make a lab03 folder in whatever directory you were in.

Part 2: Complete timer.cpp & binaryVsLinear.cpp

Take a look at timer.h and the `Timer` class—we'll use it to time program executions. Using my example from lecture, implement the class in timer.cpp so that it correctly gets the time elapsed between start() and end() calls.

Then, go to binaryVsLinear.cpp and implement linear and binary search. You should then be able to `make binaryVsLinear` and watch your binary search kick your linear search's butt with `./binaryVsLinear`. You don't have to change any part of the main() function, so your output is already in the correct format.

Part 3: Try out a bunch of different container classes

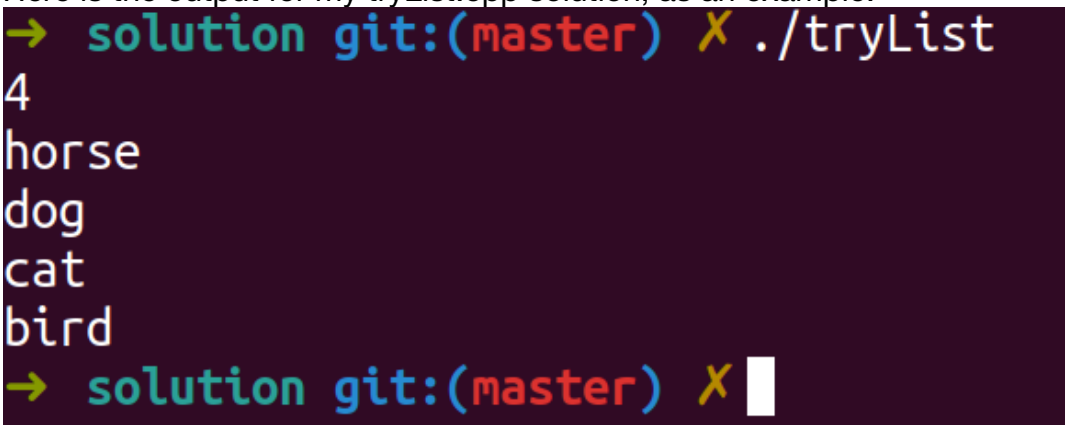
We'll be learning about a bunch of ADTs in this class, and C++ has a standard library implementation of all of them. I want you to play around with the ADTs before we cover them in depth.

You'll implement the simple programs described in the comments of tryList.cpp, tryStack.cpp, tryQueue.cpp, tryPriorityQueue.cpp, and tryMap.cpp—and I recommend that you implement the files in that order. You'll be consulting the C++ documentation every step of the way, and there are links in each .cpp file.

See the Makefile for the name of the rule for each program. For example, for tryList.cpp, it's `make tryList`.

After you get through one file, the rest should be super easy! This is mostly a way for you to learn by doing™—we'll cover these data types in excruciating detail later.

Here is the output for my tryList.cpp solution, as an example:



Part 5: Submit your code

When you're satisfied with your solutions, you can submit them to the autograder for grading. **Do not submit and assume you got 100%—you may be unpleasantly surprised. Always check your grade.**

From your solution directory, submit your code to the autograder using the following command on the terminal (**Don't copy and paste this command**—type it manually. The command is supposed to be all one line, but the pdf makes it two.):

```
turnin lab03@csci41 binaryVsLinear.cpp timer.cpp tryList.cpp tryMap.cpp
tryPriorityQueue.cpp tryQueue.cpp tryStack.cpp
```

The autograder will grade your code within a minute or so. If it's not working, please yell at Lawton to fix it. Run `view-grades` to look at your grades on the terminal, or go to the class website to view them there. You may resubmit as much as you want before the due date—just follow this same process after you've updated your programs.

Rubric

Rubric Item	**Points (23 pts total)**
binaryVsLinear.cpp runs and binary search is the clear winner	8 pts
The try_____.cpp files run and produce the correct output	15 pts (3 for each file)