CPSC 2120/2121

Lab 11

Flexible Search
Due 11:59 Fri, Nov 30

Learning Objectives

- To improve our ability to implement algorithms and data structures in C++
- To use simple data structures to implement algorithms
- To become proficient in fundamental data structures used throughout computer science
- To improve our analytical skills while gaining familiarity with mathematical tools used in the analysis of algorithms
- To implement an algorithm and understand its intricacies

Problem/Exercise

For this lab, you will implement a search program that searches a query (text), comparing it to the contents of a folder (folder will be in same directory as program and named content (make sure your approach works on Unix -- each OS has a different standard for how to find a directory/open files in it). The input will be a search query, and the output will be the closest match. This could be the name of a file in content or a line from the contents of a file in content. It may be that nothing exactly matches and you will have to try to find something that is close. Execution time from query to results displayed must be less than 5 seconds. You may assume there will be fewer than 100 files to search through, and you may assume the content directory holds only files, not subdirectories.

Use Google if you do not know how to list all files in a directory or how to read a file (e.g. http://www.fredosaurus.com/notes-cpp/io/readtextfile.html)

You are allowed to use any data structures in the C++ STL for this assignment. Your final output must be accessible through the interface provided in lab11.cpp, and must compile without errors using a makefile named Makefile (**not** provided). In order to thoroughly test your program, you will also submit a single input file named lab11.txt which has at least 3 test cases you have written to test your program.

NOTE - This should be your work. You may use existing ideas, but this should be your ideas. Copying existing code will be considered plagiarism and will result in academic integrity action being taken against you.

Additional Requirements

- You must include planning documents. While not required to be formal software engineering documents/diagrams, they must at minimum cover a description of the basic problem and discussion of what algorithm(s) you have chosen to use and why.

I/O Specification

Queries will be made in lab11.cpp. You may not change the name of the function it calls.
Queries will be strings of length at most 50.

Output should be formatted as follows
<file>
<line(s) of text for closest match> (if applicable)

For example, if the query is “frank” and the closest match is “frank.txt”, the output should be
frank.txt

(the closest match line is blank because the match is not inside the file, it is the filename itself)

If the query is “the end of all flesh is come before me”, then the output might be
Epic4.txt
The end is nigh, my doom is come before me

Or

nasa.txt
Those who have come before me

Or

panic.txt
The end of all things

Source Code Requirements

- Put a comment at the top of your source code file with your name (first and last), the date of your submission, your lab section, and the assignment's name.
- All functions should be commented. Use inline documentation as needed to explain ambiguous, tricky, or important parts of your code.
- All source code must follow good programming style standards such as properly indenting source code; and all variables, functions, and classes should be well-named.
- Your class must use dynamic memory allocation and deallocation properly, which means your class cannot contain a memory leak. You may use valgrind to check for memory leaks. Refer to the Unix manual and valgrind's online documentation for more information on valgrind.
- Since you are choosing how to define "closest match" yourself, obviously you may get an answer different than someone else for border cases. But if we have a file named "frank.txt" and search "fran", it should find the file. If we search "frank" and it doesn't find the file, your search is definitely not working.

Submission

Before the date/time stated on the CPSC 2121 Canvas webpage, you need to submit your code to handin under the correct lab assignment. Make sure to submit all of the following.

1. All source files required for this lab
2. Your testing file (lab11.txt)

After you submit, always double check that the file(s) you submitted were the correct version. To double check, download the submitted file(s), put them on one of our Unix machines, and make sure they compile and run correctly.

Grading

If your class does not compile on our Unix machines or your assignment was not submitted on time, you will receive a grade of 0 on this assignment. Otherwise, your class will be graded using the criteria below. This assignment will be graded on:

1. Your writeup, and the correctness/coherence of your justifications for each decision
2. The appropriateness of your algorithm/data structures given the decisions you made
3. The correctness of the implementation of the algorithm/data structure you chose
4. The length of time required for execution

You must test, test, and retest your code to make sure it compiles and runs correctly and efficiently on our Unix machines with any given set of valid inputs. We will only test your program with valid sets of inputs.