LAB ASSIGNMENT 8

The waiting game

The early morning fog shifts and coils outside the E.Z. Diner, tendrils of mist forming and evaporating back into the gloom at will. Seated in the booth you've come to prefer, you've forced yourself to nibble on some scrambled eggs that are presently cooling before you. Peters sits across from you, the gunmetal gray shine of her lenses revealing nothing past stoic patience. A lone silhouette, blonde hair contrasted with black suit, emerges out of the fog and drifts into the diner. As agent Bower enters, the open door momentarily allows the fog to creep inside.

Bereft of her usual appetite, she broods for a few minutes after joining the two of you in the booth. Reports had come in two nights ago, partially trustworthy sightings of a lurking presence near the trailer parks and back lots at the edge of town. Bower was of the opinion that your guy was closing in, having arrived in Whaler's End. He was now casing his future hunting grounds and immediate action had to be taken.

Tillyard had requested some additional features for the program you had helped put together, in order to better assess and compare the recordings. Bower had been promised backup in the near future, but for the time being the three of you would be on your own. Laying low was key. Random patrols wouldn't be of as much use as remaining steadfast and ready for any storms to break out. You decide that getting to work would make the time fly faster. After being promised another round of coffee by Bower, you set out on fine-tuning Tillyard's application.

For your previous assignment, add the following features:

* Replace your DynamicVector template with the STL vector. Use STL algorithms wherever possible in your application (e.g. in your filter function you could use copy\_if, count\_if). Replace all your for loops either with STL algorithms, or with the ranged-based for loop (C++11).
* Create and use a new repository to store your domain data in a text file, but not the "mylist". Do not cache the data in memory, all operations will only use files. If there is no file at the provided location create one. For this feature, use the iostream library. Create insertion and extraction operators for your entities and use these when reading/writing to files or console.
* All the functions in your application must be specified and tested. Using the [OpenCppCoverage tool](http://www.cs.ubbcluj.ro/~iuliana/oop/Tutorials/Code_coverage.pdf), make sure you have at least 99% coverage for all your modules, except for the UI. For Linux, you may use any appropriate tool you find.

All commands must be in the form:

* fileLocation fullPath (e.g. fileLocation c:\some really long\path\with spaces\myFileName.txt) - the commands must use this location from now on
* All previous commands must work with the given file (add, update, delete, etc.)

Example test run:

1. call: fileLocation c:\some really long\path\with spaces\myFileName.txt
2. call: mode A
3. call: add [valid input]
4. call: list

Check for [valid input in any format, in c:\some really long\path\with spaces\myFileName.txt]

**Note:**The server will call list just in case you use that as a trigger to save, but will only test the file contents for correctness.