Lab 2

CS321L Winter 2023, Professor Christopher Diggins

Overview

This lab will explore creating solutions with multiple projects, sharing code between projects, and using visual studio.

Tasks

A. Create a solution file named Lab2.sln
B. Add an NUnit test project named: Lab2UnitTests.csproj
C. Add a class library named: Lab2Common.csproj
D. Add the following Console projects:
    a. Lab2Type – echoes the contents of standard-in to standard-output
    b. Lab2Sort – outputs lines from standard-in, in sorted order to standard-out
    c. Lab2Find – outputs lines from standard-in that contain the specified string
E. Add a reference to Lab2Common.csproj from each console project
F. Place some of the implementation code in Lab2Common
G. Add a reference to Lab2Common.csproj from Lab2UnitTests.csproj
H. Write some unit tests that test the functionality
I. Compile and run the tests in Debug mode
J. Compile and run the tests in Release mode

About the console programs:

Lab2Type

Echoes the contents of the standard input to the standard output.

Lab2Sort

Reads lines of inputs from standard input, until there is no more input. Once there is no more input, it will output the lines in alphabetical order.

Lab2Find

Reads lines of inputs from standard input. It requires a string as the first command line argument. It will output to standard output lines of text which contain the given string.

For All programs

- If a file is passed as a command-line argument it will instead open that file as the standard-input.
- If the string "/?" is passed as a command-line argument it will output a help message

Submission

Submit a zip file containing the following:

- Solution file, projects, source files, and release mode executables
- Screen shots (.PNG) demonstrating that each program works:
    - With a /? On command line
    - With a file name on command line

- With input from standard input

How you will be graded

Total 4 points:

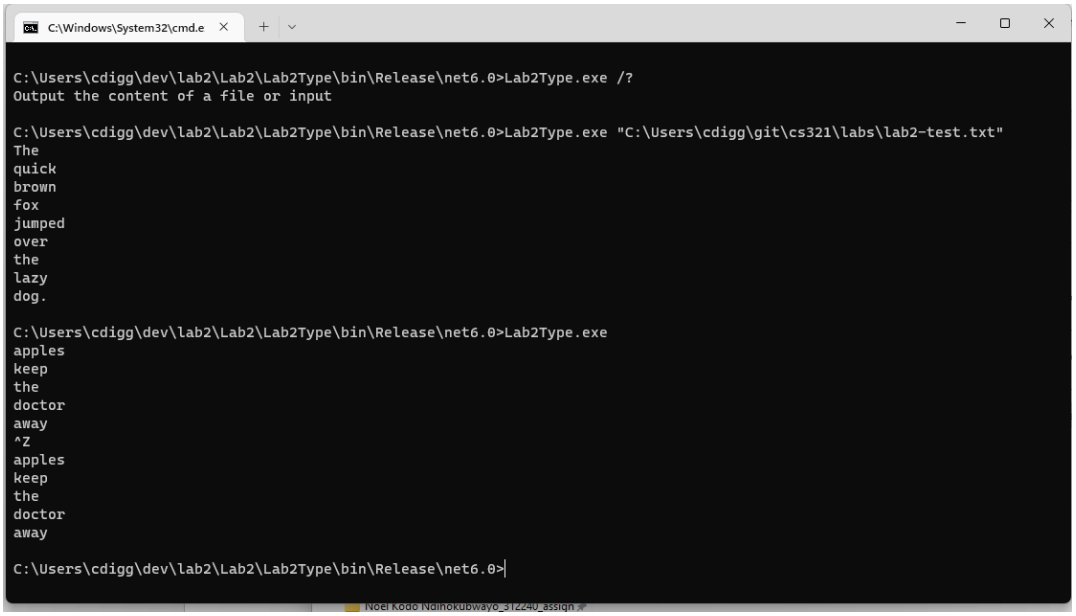
- **1 point** – screen shots show /? working for each program
- **1 point** – screen shots show passing a file name working for each program
- **1 point** – screen shots show using standard input as input
- **1 point** – you have at least one unit test, and one function in the shared project

How Specifically to Test your Program

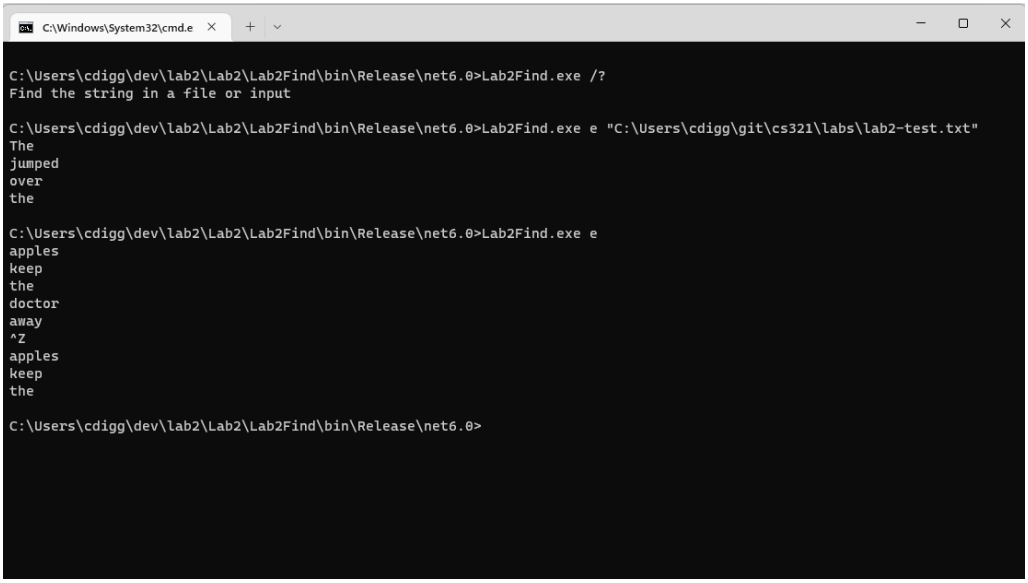
Credit to Marceline Tavernier for providing test executables for us. You can use any data you want.

Note: in these examples the program waits until all input is received before outputting the results. It is valid to also output each line as it is received without waiting for termination in the case of Lab2Type and Lab2Find.

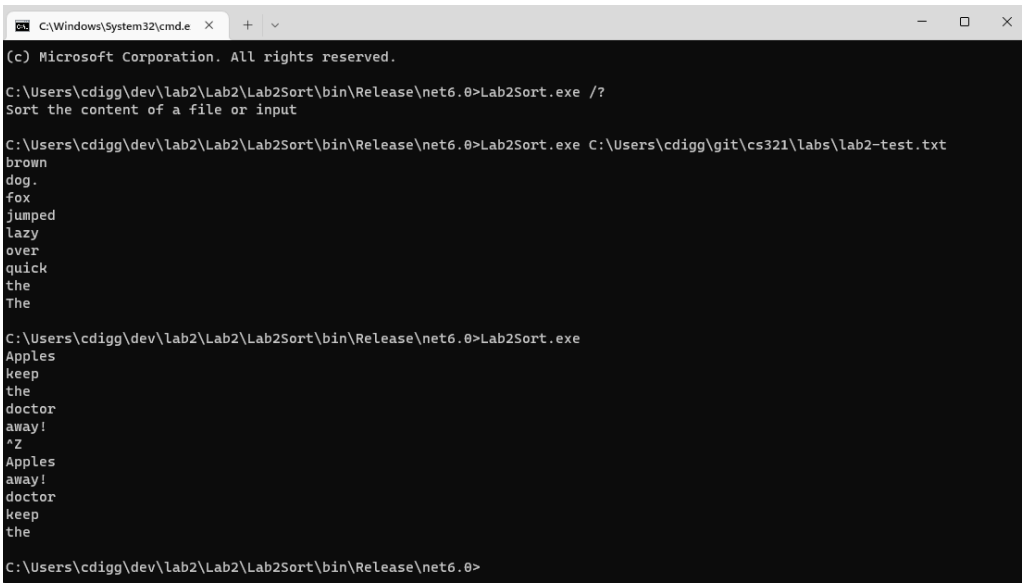
Lab2Type.exe



Lab2Find.exe



Lab2Sort.exe



Hints

- To read a line of text from standard input you can use the Console.ReadLine() function
- To write a line of text from standard output you can use the Console.WriteLine() function
- When there is no more input on the command-line the Console.ReadLine() function will return null.
- To signal that there is no more input on the command-line when manually testing your program, you can press Ctrl+Z.
- All of the text after the executable name separated by spaces are treated as command-line arguments.

- The following function can be used to redirect standard-input from a file. This means that Console.ReadLine() will now read from a file.

```
public static void SetStandardInputFromFile(string fileName)
{
    // Redirect the input from a stream reader
    var fileReader = new StreamReader(fileName);
    Console.SetIn(fileReader);
}
```

Reference Materials

- See: https://learn.microsoft.com/en-us/dotnet/api/system.console.writeline
- See: https://learn.microsoft.com/en-us/dotnet/api/system.console.readline
- See: https://learn.microsoft.com/en-us/dotnet/api/system.console.setin
- See: https://github.com/cdiggins/cs321/tree/main/code-examples/Rot13/Rot13