SET07109 Programming Fundamentals

1 Overview

This is the first coursework for SET07109 Programming Fundamentals. This coursework is worth **40%** of the coursework total. The coursework is worth **60%** of the module total. Therefore this coursework is worth **24%** of the total marks available for this module.

In this coursework you are to build a command line tool that can find a particular string in a piece of text. This command line tool will take a number of arguments which tell it how to work. For example, to find *hello* in a file `input.txt` and save the result in `output.txt` the tool will be run as follows on the command line.

```
find hello -i input.txt -o output.txt
```

The specification will go into detail about how the application should behave and the minimum requirements.

2 Specification

The `find` program is required to find a given string in some text. The program should understand the command line arguments shown in Table 1.

Argument	**Description**
`-i` *filename*	Indicates the name of the file to read the text from
`-o` *filename*	Indicates the name of the file to write the found strings to
`-c`	Indicates that the case of the string to find can be ignored

Table 1: Minimum Command Line Flags for `find` Program

Therefore we can find *hello* in a file using the command:

```
find hello -i input.txt -o output.txt
```

We can also find *hello* (ignoring case) in a file as follows:

```
find hello -i input.txt -o output.txt -c
```

Some default functionality is required by the application:

1. If no input filename is provided then the application should read from `stdin`
2. If no output filename is provided then the application should write to `stdout`

Every time the application finds the given string in the text it should output where that text is found. There are different methods you could use here (which range in difficulty):

1. Just the text searched for (e.g. if *hello* is searched for, the output file would contain `hello` repeated a number of times).
2. The line of text where the searched for string was found (e.g. if *hello* is searched for in *hello world* then the file would contain *hello world*).
3. The actual individual word(s) where the text was found (e.g. if *hell* is searched for in *hello world* then the file would contain *hello*).

3 Submission

Your code must be submitted to the Moodle Assignment Dropbox by Midnight (12am) on Sunday the 28th of February. If your submit late without prior authorisation from your personal tutor your grade will be capped at 40%.

The submission must be your own work. **If it is suspected that your submission is not your own work then you will be referred to the Academic Conduct Officer for investigation**.

All coursework must be demoed. If the coursework is not demoed to a member of the teaching team this will result in a grade of 0. The demo session is the point where the teaching team will give you feedback on your work.

The submission to Moodle must take the form of the following:

- The code file(s) required to build your application
- A collection of test files to test your application on

- A `makefile` to allow building of your application. The `makefile` must also contain a configuration that executes the test applications given above.
- A *read me* file indicating how the `makefile` is used and the toolchain used to build it (i.e. Microsoft Compiler, clang, etc.)

These files must be bundled together into a single archive (a zip file) using your matriculation number as a filename. For example, if your matriculation number is *1234* then your file should be called *1234.zip*. All submissions must be uploaded to Moodle by the time indicated.

4 Marking Scheme

The coursework marks will be divided as follows:

Description	**Marks**
File input-output functionality	2
Command line arguments work	2
Application finds strings	4
Application outputs found strings correctly	4
Default application behaviour works correctly	2
Resources freed and cleared up correctly	2
Code quality	2
Correct use of other necessary data types (e.g. `struct`, `enum`)	2
Makefile works correctly	2
Submission zip folder complete and correctly collated	2
Total	24

To pass this coursework you will require a minimum of 10 marks, although you should be aiming to achieve a much higher mark to illustrate your understanding of the material.

BE WARNED! This coursework requires an understanding of the material covered in the first 5 units of the module. If you do not complete these you will struggle. DO NOT LEAVE THIS WORK TO THE LAST MINUTE!

If you have any queries please contact a member of the teaching team as a matter of urgency. This coursework is designed to be challenging and will require you to spend time developing the solution.