

1 Introduction

These exercises are all dealing with C++ decisions and repetition structures. Each exercise should be its own separate project. When you are ready to submit your work create a folder called `Homework03` in that folder have separate folders for each project, one folder per project. Put all the code files needed for that project in its respective folder. Do not include the files that the IDE creates, I just want the code files. Zip the entire `Homework03` folder up into a single zip file and submit it.

Remember to follow the coding and documentation standards for the class listed on the MyClasses page.

2 Exercises

1. Write a program that will read in from a file integer numbers and determine the minimum and maximum of those numbers. The numbers in the file are no larger than a long data type but they can be positive or negative. Remember when finding the minimum and maximum of a list of numbers that you know nothing about you want to set the first number in the list to both the maximum and the minimum, then process the rest of the file. Two files have been provided to you for testing `numbers.txt` and `numbers2.txt`. Two runs are below.

```
Input the input filename: numbers.txt
Maximum = 2147481629
Minimum = -2147482576
```

```
Input the input filename: numbers2.txt
Maximum = 2140019365
Minimum = -2145851460
```

2. Write a program that will take a filename from the user and a target string that can either be a single word or a phrase of several words. The program is to find all of the occurrences of the input word or phrase in the file. The searching must be case insensitive. Also, keep in mind that the word or phrase could be present several times on one line. If the phrase is broken up between two lines in the file you do not need to worry about counting it. I have included two text files for this exercise, `DATHT.txt` and `HGGT.txt`. These files contain the Hitchhikers Guide to the Galaxy trilogy, which is actually 5 books. In the file `DATHT.txt` the text is broken into over 26,000 lines as one would expect to read and the file `HGGT.txt` is the same words but is on a single line. Your program must work for both files without alteration. An example run is below.

```
Input the input filename: DATHT.txt
Input the word or phrase to be found: Arthur

The file contains 1576 occurrences of "Arthur"
```

In the Hitchhikers Guide to the Galaxy the main characters are Arthur Dent, Ford Prefect, Zaphod Beeblebrox, Trillian, Marvin, and Slartibartfast. Do searches on both files for Arthur, Arthur Dent, Ford, Zaphod, Beeblebrox, Zaphod Beeblebrox, Trillian, Marvin, and Slartibartfast and report the number of occurrences of each.

Put the results of these searches in a text file and include them with your code submission. You can create a text file with a number of applications, if you are on Windows I would suggest Notepad or the freeware application Notepad++, on the Mac there is TextWrangler also free at the app store.

3. Write a program that does a simple Caesar cipher on an input file.

In cryptography, a Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet.¹

So with a shift of 3 the letter A would be replaced with D, B with E, C with F, and so on. We would also wrap around at the end of the alphabet. So W would be replaced with Z, X with A, Y with B, and Z with C. To decrypt you shift in the other direction. So again with a shift of 3, we would decrypt D with A, E with B and so on.

Write a program that will take in a filename from the user, a shift between 0 and 25, and either E or D to indicate encryption mode or decryption mode. Then have the program either encrypt or decrypt the message with the desired shift. Error check the range of the shift and error check the input of E or D but allow the user to input either uppercase or lowercase on the mode. Also check that the file exists, if it does not then have the program ask for another filename.

The program should take the file contents along with the encryption or decryption options, convert it to all uppercase and remove any characters that are not alphabetic characters, then encrypt or decrypt the message. The message is to be output to another file, the input file is not to be altered. The output filename is to be the input filename with ".out.txt" appended to the end. So for an input file named CipherInputFile.txt the output file is CipherInputFile.txt.out.txt.

One hint here is if you take each character of the message and find its distance from A, add the shift modulo 26 for encryption and add that onto a character A then you have the encrypted character. Same goes for decryption except that you subtract the shift. A run of all correct inputs is below.

```
Input the input filename: CipherInputFile.txt
Input the shift (0 - 25): 4
Input E to encrypt and D to decrypt: e
```

A run that has input errors is below.

¹https://en.wikipedia.org/wiki/Caesar_cipher

```

Input the input filename: input.txt
Invalid filename, file input.txt not found.
Input the input filename: badfile
Invalid filename, file badfile not found.
Input the input filename: badfile.txt
Invalid filename, file badfile.txt not found.
Input the input filename: CipherInputFile.txt
Input the shift (0 - 25): 64
Invalid input, please input a number (0 - 25).
Input the shift (0 - 25): -3
Invalid input, please input a number (0 - 25).
Input the shift (0 - 25): 5
Input E to encrypt and D to decrypt: r
Invalid input, please input E or D.
Input E to encrypt and D to decrypt: e

```

The CipherInputFile.txt file, that is included with the files for this homework, contents are below.

In cryptography, a Caesar cipher, also known as Caesar's cipher, the shift cipher, Caesar's code or Caesar shift, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plaintext is replaced by a letter some fixed number of positions down the alphabet.

After processing an encryption with shift 4 the output file CipherInputFile.txt.out.txt looks like the following, note that this can be a single line file, the text does not need to break here or at all.

```

MRGVCTXSKVETLCEGEIWEVGMTLIVEPWSORSAREWGEIWEVWGMTLIVXLIWLMJXGMTLIVGEIWE
VWGSHISVGEIWEVWLMJXMWSRISJXLIWMQTPIWXRHQSWXAMHIPCORSARIRGVCTXMSRXIGLR
MUYIWMXMWEXCTISJWYFWXMXYSRGMTLIVMRALMGLIEGLPIXXIVMRXLITPEMRXIBXMWVIT
PEGIHFCEPIXXIVWSQIJMBIHRYQFIVSJTSWMXMSRWHSARXLIPTLEFIX

```

After processing a decryption with shift 4 on the input file CipherInputFile.txt.out.txt the output file CipherInputFile.txt.out.txt.out.txt looks like the following, note that this can also be a single line file, the text does not need to break here or at all.

```

INCRYPTOGRAPHYACAESARCIPHERALSOKNOWNASCAESARSCIPHERTHESHIFTCPIPHERCAESA
RSCODEORCAESARSHIFTISONEOFTHE SIMPLESTANDMOSTWIDELYKNOWN ENCRYPTIONTECHN
IQUESITISATYPEOFSUBSTITUTIONCIPHERINWHICHEACHLETTERINTHEPLAINTEXTISREP
LACEDBYALETTERSOMEFIXEDNUMBEROFPOSITIONSDOWNTHEALPHABET

```

3 Submit Your Work

When you are ready to submit your work create a folder called Homework03 in that folder have separate folders for each project, one folder per project. Put all the code files needed for that project in its respective folder. Do not include the files that the IDE creates, I just want the code files. Zip the entire Homework03 folder up into a single zip file and submit it.