This lab exercise is a continuation of Lab 12.1. If you have not completed Lab 12.1, you should work on that before beginning Lab 12.2.

Optional Lab:

Because this lab is rather difficult, we are making it an **optional** lab exercise. If you complete this Lab Exercise, then it will help your grade slightly, but if you do NOT complete it, that will NOT reduce your course grade.

Background Information

In Lab 12.1, we wrote a program to create a "Listing File" of an input text file. That is, we created an output text file with extra information about the input file:

Each line of text in the output file contains information about the corresponding line of text in the input file, in the format specified below:

```
lineNumber < fileOffset, length >: contents
```

Notice that each line of original (input) text is preceded by:

- The line number.
- The file offset (bytes from the beginning of the file).
- Length of the original text for that particular line.

Sample Output File (from Lab 12.1)

```
lineNumber < fileOffset, length >: contents
```

```
1<0,57>: Four score and seven years ago our fathers brought forth
2<59,54>: on this continent a new nation, conceived in liberty,
3<115,66>: and dedicated to the proposition that all men are created equal.
4<183,70>: Now we are engaged in a great civil war, testing whether that nation,
5<255,63>: or any nation so conceived and so dedicated, can long endure.
6<320,48>: We are met on a great battle-field of that war.
7<370,75>: We have come to dedicate a portion of that field, as a final resting place
8<447,66>: for those who here gave their lives that that nation might live.
9<515,61>: It is altogether fitting and proper that we should do this.
10<578,68>: But, in a larger sense, we can not dedicate, we can not consecrate,
11<648,64>: we can not hallow this ground.  The brave men, living and dead,
12<714,41>: who struggled here, have consecrated it,
13<757,45>: far above our poor power to add or detract.
14<804,64>: The world will little note, nor long remember what we say here,
15<870,45>: but it can never forget what they did here.
16<917,77>: It is for us the living, rather, to be dedicated here to the unfinished work
17<996,61>: which they who fought here have thus far so nobly advanced.
18<1059,78>: It is rather for us to be here dedicated to the great task remaining before us
19<1139,72>: --that from these honored dead we take increased devotion to that cause
20<1213,55>:   for which they gave the last full measure of devotion
21<1270,73>: --that we here highly resolve that these dead shall not have died in vain
22<1345,64>: --that this nation, under God, shall have a new birth of freedom
23<1411,68>: --and that government of the people, by the people, for the people,
24<1481,34>:   shall not perish from the earth.
25<1517,15>: Abraham Lincoln
```

Random-Access File I/O

In this lab exercise, we write an interactive tool to read and modify a file through "random access" operations. (Feel free to re-use portions of old labs, as appropriate, for the interactive user interface processing.) Because the program we write is a somewhat clumsy tool for modifying a file, we use the output from Lab 12.1 as a guide, to assist the user in finding the desired file offset values.

Programming Exercise

Write an interactive program which supports the following commands:

b	open file for input AND output, in **BINARY** mode.
c	Close input/output file.
f	Flush the output buffer to the actual file
h	print help text.
q	quit (end the program).
r	random-access read from input/output file.
t	open file for input AND output, in **TEXT** mode.
w	random-access write part of input/output file.

The Random-access Read (r) Command:

This function should ask the user for a file offset to begin the read, and for the number of bytes to read. The bytes should be stored in a character array.

After the data is stored in the array, display it one byte at a time according to the following rules:

- If the byte contains a space character, display it as a decimal integer.
- Otherwise, if the byte contains a "printable" character, display it as a single character, surrounded by single quote characters.
- Otherwise, display the character as a decimal integer.

Recall that we discussed library functions for testing characters when we covered Chapter 10 in the textbook.

The Random-access Write (w) Command:

This function should ask the user for a file offset for writing, and for the replacement text which will **overwrite** the existing bytes of the file. (Whatever bytes were previously in the file, at the specified locations, are destroyed.)

The Flush (f) Command:

When your program executes a write function, the data is written to an output buffer (in memory). The actual file does not normally get modified until some amount of data has been written to this buffer. If your program wishes to force the **fstream** object to write to the actual file, it can call the fstream "**flush()**" function.

Text Mode vs. Binary Mode

There are two commands for opening the data file: **t** for text mode, and **b** for binary mode.

What difference do you observe in the output of the **r** command, between the two different modes? (Refer to *Appendix A* in the textbook for assistance in understanding the difference.) **NOTE**: If you are running on a UNIX-based computer, such as a Macintosh, you may observe no difference between text mode and binary mode.

Sample Interactive Sessions

In these examples, what the user types is shown in **bold**. In actuality, what the user types would appear as the same text format as the output.

Sample Interactive Session: Binary Mode

```
Enter command (or 'h' for help): h
Supported commands:
        b               open file for input AND output, in BINARY mode.
        c               Close input/output file.
        f               Flush output buffer to the actual file.
        h               print this help text.
        r               random-access read.
        t               open file for input AND output, in TEXT mode.
        q               quit (end the program).
        v               Verbose Mode ON/OFF.  (optional debug aid)
        w               write(random-access write) part of input / output file.

Enter command (or 'h' for help): b
Enter name of input/output file: Gettysburg.txt
File open (binary mode) successful:  Gettysburg.txt
Enter command (or 'h' for help): r
Enter file offset for random-access read: 500
Enter number of bytes to read: 20
'm' 'i' 'g' 'h' 't' 32 'l' 'i' 'v' 'e' '.' 32 32 13 10 'I' 't' 32 'i' 's'
Enter command (or 'h' for help): w
Enter file offset for random-access write: 500
Enter text to overwrite portion of file: COULD
Enter command (or 'h' for help): r
Enter file offset for random-access read: 500
Enter number of bytes to read: 20
'C' 'O' 'U' 'L' 'D' 32 'l' 'i' 'v' 'e' '.' 32 32 13 10 'I' 't' 32 'i' 's'
Enter command (or 'h' for help): w
Enter file offset for random-access write: 500
Enter text to overwrite portion of file: should
Enter command (or 'h' for help): r
Enter file offset for random-access read: 500
Enter number of bytes to read: 20
's' 'h' 'o' 'u' 'l' 'd' 'l' 'i' 'v' 'e' '.' 32 32 13 10 'I' 't' 32 'i' 's'
Enter command (or 'h' for help): q
```

Sample Interactive Session: Text Mode

```
Enter command (or 'h' for help): h
Supported commands:
        b               open file for input AND output, in BINARY mode.
        c               Close input/output file.
        f               Flush output buffer to the actual file.
        h               print this help text.
        r               random-access read.
        t               open file for input AND output, in TEXT mode.
        q               quit (end the program).
        v               Verbose Mode ON/OFF.
        w               write(random-access write) part of input / output file.

Enter command (or 'h' for help): t
Enter name of input/output file: Gettysburg.txt
File open (text mode) successful:  Gettysburg.txt
Enter command (or 'h' for help): r
Enter file offset for random-access read: 500
Enter number of bytes to read: 20
's' 'h' 'o' 'u' 'l' 'd' 'l' 'i' 'v' 'e' '.' 32 32 10 'I' 't' 32 'i' 's' 32
Enter command (or 'h' for help): w
Enter file offset for random-access write: 510
Enter text to overwrite portion of file: 012345
Enter command (or 'h' for help): r
Enter file offset for random-access read: 500
Enter number of bytes to read: 20
's' 'h' 'o' 'u' 'l' 'd' 'l' 'i' 'v' 'e' '0' '1' '2' '3' '4' '5' 't' 32 'i' 's'
Enter command (or 'h' for help): q
Press any key to continue . . .
```

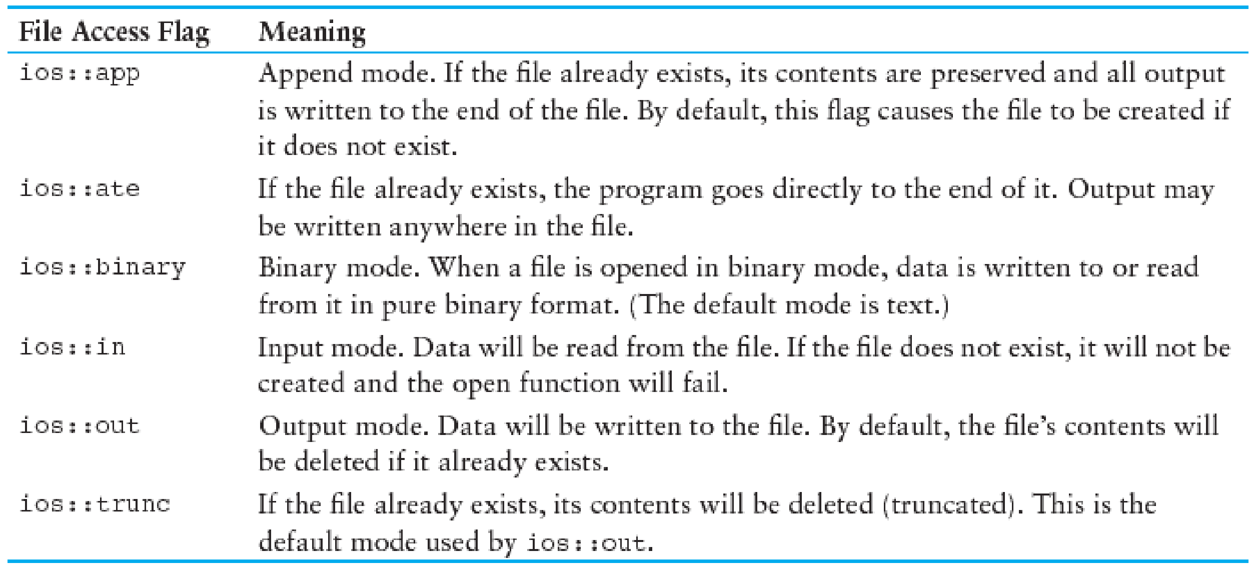
Look at the file contents by opening it with a text editor, such as *NotePad*. What difference do you see between the latest version of the data file (**Gettysburg.txt**) and its "ORIGINAL" version (**Gettysburg_ORIGINAL.txt**)?

Technical Reminders

In the lecture that covered "*Advanced File Operations*" (Chapter 12), we discussed a few technical details needed to complete this lab:

- When opening a file for both input and output, you must use a **`fstream`** object, with appropriate file access flag settings. (See section 12.10, pages 705-709.)

Table 12-2



File Access Flag	Meaning
`ios::app`	Append mode. If the file already exists, its contents are preserved and all output is written to the end of the file. By default, this flag causes the file to be created if it does not exist.
`ios::ate`	If the file already exists, the program goes directly to the end of it. Output may be written anywhere in the file.
`ios::binary`	Binary mode. When a file is opened in binary mode, data is written to or read from it in pure binary format. (The default mode is text.)
`ios::in`	Input mode. Data will be read from the file. If the file does not exist, it will not be created and the open function will fail.
`ios::out`	Output mode. Data will be written to the file. By default, the file's contents will be deleted if it already exists.
`ios::trunc`	If the file already exists, its contents will be deleted (truncated). This is the default mode used by `ios::out`.

- When performing input / output on **binary** files, the functions **read()** and **write()** must be used. (See section 12.7, pages 688-693.)
- When performing **random access** input / output on a file, use the **seekg()** and **seekp()** functions to change the read position or write position for the file. The **tellg()** and **tellp()** functions return the "current" read position or write position. (See section 12.9, pages 697-705.)