The University of Alabama in Huntsville
Electrical and Computer Engineering
Project 8 (30 points) – 10% bonus for exact match of all program output
Submit Your Solution Using Canvas by 10PM Friday October 19, 2018
(A late submission drop box will be available on 10/19/2018 from 10PM to 11:59PM)

<Project 8 Description>

You will write a program that loops until the user selects 0 to exit. In the loop the user interactively selects a menu choice to compress or decompress a file. There are three menu options:
Option 0: allows the user to exit the program.
Option 1: allows the user to compress the specified input file and store the result in an output file.
Option 2: allows the user to decompress the specified input file and store the result in an output file.

Assume that the user types any character when prompted to select a menu option. If an input file cannot be opened, the program prints an error message and reprints the menu as shown in the sample solution. The output file is assumed to open correctly.
In addition to your main function*, *Your solution must include the following non-trivial functions:

1. A function to print out the menu – a void function. Output statements only
2. A function that returns any integer value entered by the user
3. A function to open an input file[1] – may want to use as a Boolean value returning function
4. A function to open an output file[1] **(this is a very basic function – not testing required)**
5. A function for compression[2]
6. A function for decompression[2]

[1]These functions require 1 reference parameter – an input file stream or output file stream variable. These functions must be used to open the files.
[2]These functions require 2 reference parameters – an input and an output file stream variable

Example ***The compression scheme is illustrated below.***

Decompressed Text	Compressed Text	Observations
`abbccc\n`	`1a2b3c2\n`	`1a` ➔ `1 copy of a, 2b` ➔ `2 copies of b,…`
`\n`	`4d1\n`	`2\n` ➔ `2 copies of \n and so on`
`dddd\n`		

Note #1: \n represents a newline character

Note #2: You should be able to take a file compressed by your program and use your program to decompress it and recover the original file. In other words, if Option 1 compresses X to produce the file Y, then Option 2 should decompress Y to produce a file Z that is identical to X.

Reminder:
Your solution will be tested with several compressed and decompressed files so make sure you test your solution thoroughly before submitting it for grading. Run the sample solution to determine how the program operates.

Your program's output format should match the output produced by the provided sample solution. You may run the sample solution **Project_08_solution** by typing the following at a command prompt in a terminal window**:**

/home/work/cpe211/Executables/Project_08/Project_08_solution
Input files are contained in P8_in.zip. Read the README.TXT file included in the zip file

Note: make sure that all necessary inputs are available in the directory that the terminal window is in when the program solution is run (i.e. your ~/CPE211_FALL18/Project_08 directory). Also, run the comparison script before submitting your program to verify your output versus the sample solution output

/home/work/cpe211data/Project_08/CompareSolution.bash Project_08.cpp

<Project 8 Requirements and Restrictions>

- **Global variables are not allowed.**
- **Using global variables will result in a score of 0 on this assignment.**
- **You may only use concepts presented in chapters 1-9 of your textbook.**
- **You must use function prototypes and all function definitions go below main ← ← ←**
- **The program must have the six non-trivial functions listed on page 1. ←**
- **The Print Menu Function will have output statements only. It will not read in any value.**
- **The function to obtain an integer is to return any integer value. This function will handle the case of invalid character entry**
- **The menu selection is to be read into an integer variable – not a character**
- **Recursive function calls are not allowed and all user defined functions calls are to be made from main. User defined functions cannot call other user defined functions ← ←**

<Project 8 Directions>

Using your favorite text editor, type your solution and save it as a file named **Program_09.cpp** within your **CPE211_FALL18/Project_08** directory. If there are syntax errors, correct them and compile again. Once your program successfully compiles, run it and verify that the output for your program matches the output from the provided solution executable –

Once you are satisfied with your solution, submit Project_08.cpp via Canvas.

NOTE: make sure that you do not change the order in which the information is entered. An automatic script is used to process all lab submissions, and if the order of the input information is modified, the script will not work properly with your program.

<Project 8 Help, Hints and Concepts>

There are 6 obvious tasks that need to be done, and these tasks are written as functions: Print a menu, obtain an integer value, open an input file, open an output file, compress a file and decompress a file

The compression and decompressing technique used in this lab is a very basic idea that lends itself to using loops. In this project, newline characters are included in the compressed file and are considered to be just another character.

Compression algorithm description: The characters in the file to be compressed are read (using the get function) one character at a time. If a character repeats, then the number of times the character is repeated is followed by only one of the characters. Therefore, if a string of **11 P's (PPPPPPPPPPP)** occurs in the input information, the compressed file has the value **11P**.

When reading the file for compression, use the get function to read each character because spaces, tabs and new line characters will be read.

Decompression algorithm description: The decompression technique used is easy (For this project, assume the output file contains no errors or missing information) to implement. Read in an integer value (using the extraction operator), then use the get function to read in the character that follows. The character read is written to the output file the number of times specified by the integer value read. For example, **5T** is decompressed as **TTTTT.** Remember you have to read new line characters, so **the get function must be used** to read the character from the input file**. Also, the compressed file will not contain compressed digits since there is no easy way to obtain the number of times the digit occurs** (i.e. processing 25 – two number fives and 115 – eleven number fives).

When reading the file for decompression, use the extraction operator for reading the integer value and the get function (so that whitespace characters can be read) to read the character.

close function: This function is used to terminate the association of a file with a file stream variable. **someStream.close()**: breaks the connection between the file stream variable **someStream** and the physical file on disk.

clear function: This function is used to take a file stream out of the fail state mode. The use of this function is: **someStream.clear()**: . To completely reset a file stream variable for use with another file, the file stream variable needs to be closed and cleared. **The way to reset a file stream variable** (**someStream** in this handout) is shown below (the order is important):

someStream.close():
someStream.clear():

Open the input file and test the status of the input file stream. **If the input file was not successfully opened, print out an error message and abort the.** If the input file is successfully opened, continue on with opening an output file – assume it opens successfully and performing the selected operation. (Remember to test the sample solution for program operation involving an invalid input file).

Each time a menu selection is made to compress or decompress a file, an input file and an output file must be opened and associated with an appropriate file stream variable.

After each compression or decompression of a file, the input and output file stream variables need to be reset (see the above discussion with the close and clear functions).

Run the provided sample input files to get an idea of how the compression and decompression algorithms for this project works.

Remember that file streams must be reference parameters in a function.

Compressing the file will require loops that keep track of a previous value, and the file to be compressed will have to be read one character at a time.

For printing the menu and menu selection, the program is required to use two functions. Use one function to print the menu and one to obtain an integer value for the selection. Using two functions is easier to follow and code – and it prevents recursive function calls.

```
PrintMenu();
selection = GetInteger(); // returns any integer value
// process integer value returned – may not be a valid menu choice of 0,1 or 2
```

The program will be easier to write if the priming read concept is used where the menu is printed and an integer value for the menu selection is obtained before a loop. Then at the end of the loop, the menu is printed and another selection is obtained. The loop in question is the one that keeps processing user commands until the exit option is selected.

Write a function that continually prompts for an integer menu choice until an integer value is entered. Remember that the input stream goes into the fail state for a non-integer character.

The function used to obtain an integer is allowed to call the print menu function – but only in the loop handling an invalid character entry ← ←

An empty input file for compression or decompression results in a message written to the terminal, and the output file will be empty.

For compression, make sure that your program outputs the last compressed character information – Compare your last output with the sample solution. The last number-character pair may not be printed out with some methods

Make sure your program works for the compression input file P8_com_4.txt. This input file has multiple blank lines to end the file. Your program may go into an infinite loop for this input file. To prevent the infinite loop, you need to modify one of your tests to include the status of the input stream as well. For example, you may have in your code:

```
while (currChar == prevChar)
{
}
```

this needs to become:

```
while (currChar == prevChar && inputFileStream.good())
```

where inputFileStream is the name of your input file stream variable.

Compression is harder to program than decompression.

<Project 8 Algorithm(s)/Functional Decomposition/Outline>

The following algorithms/Functional Decomposition layout should help with writing your program:

```
Put function prototypes above main
Declare variables in main
Print the menu (Use a function to print out the menu)
Obtain an integer choice (Use a function to obtain an integer)
Loop while the choice is not 0
        if the choice is compress or decompress (1 or 2)
                Open the input file (use a function)
                if file stream is in the fail state mode
                        print out an error message – cannot open file
                else If compression selected
                        Open the output file (use a function)
                        Compress the input file (use a function)
                else if decompression selected
                        Open the output file (use a function)
                        Decompress the input file (use a function)
        else
                Error message for invalid integer entered

        Reset the input and output file stream variables using
        filestream.close(); filestream.clear()

        Print the menu
        Obtain integer choice
End of loop
Output program ending message
End of program
```

This is one way to handle the opening of the files. If you want, you can have something like:

```
If (compress chosen)
  Open input file
  If in fail state
    Open file error message
  Else
    Open the output file
    Compress file
Else if (decompress chosen)
   Same steps as for compress
Else – invalid integer error
message
```

Function Definitions go after main

Functions for opening an input or output file, prompt the user for the filename, read the name, echo print it and then open the file. These functions have one reference parameter – the file stream being opened. Write one function for the input file and one for the output file

Write a function that prints the menu

Function for reading an integer

Read using extraction operator the “integer” value entered by user
Loop while cin is in the fail state
Reset the file stream
Read in a single character with the extraction operator
Echo print the character read
Remove unwanted characters from the input stream(ignore function)
Print out an error message
Print out menu – use the function (only place this function call can occur in this function)
Read in the next “integer” value (this is not a function call here)
End of loop – loop is terminated when an integer is entered
Echo print the integer value entered by the user
Remove any possible unwanted characters from the input stream

For decompression and compression functions:

- Perform a priming read before entering a loop to perform the decompression or compression.
- If priming read puts the stream into the fail state, print out empty file message and return
- Otherwise proceed with the compression or decompression algorithm

Hint: Write the program and functions leaving the compression and decompression functions for last. Verify that everything else works before writing these last 2 functions.