Project 8
Functions

- In Project 8, you will utilize functions to compress or decompress an input file.
  - To compress the file, characters are read one at a time from the input file. All characters are replaced by a number-character pair.
  - To decompress a file, a number-character pairing is read and then expanded to just the characters represented by the number-character pair
  - Output of the compression or decompression is written to an output file.

Project 8 Constraints

- Use only material from Chapters 1-9 of your textbook
- Use at least the 5/6 non-trivial functions mentioned in the description

- ***Echo print all inputs read using cin***
- ***No Global Variables allowed***
- ***Menu selection is read into an integer***
- ***Read restriction in the description***

Project 8 information

- A menu is printed with a list of three options
- The users selection is read and processed – assume that any character may be entered
- The user decides to (0) Exit the program (1)Compress a file, or (2) Decompress a file
  - Integer values other than 0,1 or 2 result in an **invalid integer error message** printed to the terminal
  - Non-digit characters result in an **invalid character error message** printed to the terminal
  - User selection values are echo printed to the terminal

Project 8 information Continued

- For compression, read the input file character by character and create number-character pairs
  - AAABBBCDE compresses to 3A3B1C1D1E
  - New line characters are included in this as well and show up in a compressed file as just a number at the end of the line – If the above line was terminated with a new line character the output is 3A3B1C1D1E1
-

Project 8 information Continued

For decompression, the input file is read on a number-character pair basis

- – 4A2B1C decompresses to AAAABBC
- – Numbers will not be decompressed
  - Difficult to decompress correctly
  - Compressing 222AAA gives 323A, decompressing 323A would be 323 A’s

- Opening the input file
  - – If the opening of the input file fails,
    - Print out an error message and terminate the current selection being processed.

Project 8 information Continued

- For processing the menu choice, the program must handle invalid character entry (i.e. user enters in a letter instead of an integer)

- Each time compressing or decompressing a file is selected,
  - – an input file is opened, (test for successful open)
  - – an output file is opened, (assume that it opens)
  - – The input file is processed and output written to the output file.
  - – The file streams for input and output are reset

Project 8 information continued

- An empty input file results in an empty output file and a message to the terminal

- **Read the entire handout – there is some useful information in the Help, Hints and concepts section.**

- *Verify operation with invalid input file names and invalid (out of range integer values and characters) for menu choices*

- ***Sample Solution and comparison script***

/home/work/cpe211/Executables/Project_08/Project_08_solution

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

Hints

- Use 5 or 6 functions – print menu, open input file, open output file, compress a file, decompress a file, obtain an integer.

- Process valid menu selections by using a loop that processes an integer entered and terminates when exit is selected

- Before entering the loop, create a priming read of a menu selection by printing the menu and obtaining the users first choice

Hints Continued

- Inside the loop, process the user selection (an integer value) using if-then-else-if statements or a switch statement
- Look at the P8_error page for information on error messages
- At the end of the loop
  - Reset the input and output file stream variables (see the next slide).
  - print the menu and
  - obtain another menu selection

Hints Continued

- To reset the input and output file streams use

  filestream.close();

  filestream.clear();

  Where filestream is the name of your file stream variable
- To error proof the menu choice:
  - Use a function that reads in an integer value
    - Prompt for an integer
    - Read in the integer entered
    - While the input stream is in the fail state
      - Reset the input stream (cin) by using cin.clear();
      - Use extraction to read a single character and then echo print it
      - Remove the remaining invalid character(s) from cin – use ignore
      - Prompt for an integer
      - Read in the integer entered
    - Return the integer value entered

Partial Functional Decomposition

- Print menu
- Obtain integer value
- While integer value is not 0 (exit choice)
  - Process selection
    - compress,
      - Open input file, output file, then compress
      - Or print out **Compression function is under development**
    - decompress
      - Open input file, output file, then decompress
    - invalid integer choice – error message
  - Close and clear the input and output file streams
  - Print menu and obtain next integer value to process

Menu selection Method #1

(while loop method)

```
Print the menu  // (use a function) this line and the next act like priming read
Obtain an integer // (use a function and the technique on slide 10)
While the integer read is not 0
    if value is 1, perform compression steps
    else If value is 2 perform decompression steps
    else output error message for invalid integer provided
    close and clear the input and output file streams
    Print the menu
    Obtain an integer value
End of while loop
Output message that exit was selected
End of program
```

Menu selection Method #2

(do-while loop method)

```
Start a do while loop
    Print the menu  // (use a function) this line and the next act like priming read
    Obtain an integer (use a function and the technique on slide 10)
    if integer read is 0, output exit message
    else If integer read is 1, perform compression steps
    else If value is 2 perform decompression steps
    else output error message for invalid integer provided
    close and clear the input and output file streams
End of do-while loop that continues while integer entered is not 0 (or
   conversely exits when integer entered is 0)
```