The University of Alabama in Huntsville
Electrical and Computer Engineering
Project 9 (50 points) – 10% bonus for exact match of all program output

Submit Your Solution Using Angel by Noon, Friday March 15, 2013

(A late submission drop box will be available on 03/15/13 from Noon to 2pm)

<Project 9 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 process the number of consonant or vowels in a file.
There are three menu options:

Option 0: allows the user to exit the program.
Option 1: Determine the number of consonants and lines in a file
Option 2: Determine the number of vowels and lines in a 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. Output is to the terminal

In addition to main*, *your solution must include the following non-trivial functions:

1. A function to print out the menu
2. A function that returns an integer value entered by the user (for menu choice)
3. A function to open an input file[1] – may want to use as a Boolean value returning function
4. A function for processing the input file – can use one function to process both vowels and consonants, or can have two separate functions.

[1]**This function requires at least one reference parameter – an input file stream variable. A function must be used to open the input file.**

Your program's output format should match the output produced by the provided sample solution. You may run the sample solution **Project_09_solution** by typing the following at a command prompt in a terminal window**:**

/home/work/cpe112/Executables/Project_09/Project_09_solution
Input files are contained in P9_in.zip.

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 ~/CPE112_SPR13/Project_09 directory). Also, run the comparison script before submitting your program to verify your output versus the sample solution output

/home/work/cpe112data/Project_09/CompareSolution.bash Project_09.cpp

<Project 9 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 four/five non-trivial functions listed above ← ←**

<Project 9 Directions>

Using your favorite text editor, type your solution and save it as a file named **Program_09.cpp** within your **CPE112_SPR13/Project_09** 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_09.cpp via Angel.

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 9 Help, Hints and Concepts>

There are 4 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, and process a file for consonants or vowels (may want to use two separate functions – one for each action)

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.** The program then proceeds with reprinting the menu and reading the next selection at the end of the loop. If the input file is successfully opened, continue on with 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 process vowels or consonants, an input file must be opened and associated with an appropriate file stream variable.

After each processing of a file, the input file stream variable needs to be reset – done at the end of the loop before reprinting the menu and obtaining the next selection

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). This applies for Dev-C++ or when g++ is used to compile the program:

```
someStream.close();
someStream.clear();
```

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

For the menu and menu selection, it is recommended that you 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 (when a function calls itself).

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

Your program will be easier to write if you use the priming read concept, and perform a read of a menu selection before a loop. You will then need to print the menu and read a menu selection at the end of that loop. 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.

An empty input file results in a file empty message being printed to the terminal

Messages written to the terminal are for the following problems: file open error, invalid integer value, invalid character and empty input file

Run the sample solution to understand the program operation and what is expected for output

<Project 9 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 to exit
        if the choice is vowels or consonants (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 vowels selected
                        Process the file for vowels (use a function)
                else if consonants selected
                       Process the file for consonants (use a function)
                else
                     Error message for invalid integer entered

        Reset the input file stream variable using
        filestream.close(); filestream.clear();

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

Function Definitions go after main

Function for opening an input file - prompt the user for the filename, read the name, echo print it and then open the file. This function has at least one reference parameter – the file stream being opened. Another possible reference parameter is the name of the file being opened.

Function for printing the menu – series of output statements

Function for reading an integer (shown for a while loop, but a do-while loop can be used as well)

Read using extraction operator the "integer" value entered by the user
Loop while cin is in the fail state
Reset the file stream cin
Read in a single character with the extraction operator
Echo print the character read
Remove unwanted characters from the input stream ←
Print out an error message
Print the menu (function call)
Read in the next "integer" value
End of loop – loop is exited when an integer is entered
Echo print the integer value entered by the user
Remove any possible unwanted characters from the input stream ←

For processing the file (same steps apply for vowels or consonants):

- Initialize loop counters
- Perform a priming read of a single character before entering an **EOF** loop to process the file.
- If priming read puts the stream into the fail state, print out empty file message and return
- Enter loop and process the character read – is it a vowel, consonant or new line character. Can use the functions tolower(char) and/or isalpha(char) to help. These functions are defined in the cctype header file.
    - For tolower: chLower = tolower(ch); // returns the lower case of ch if ch is a letter otherwise it returns the character contained in ch (converts uppercase to lowercase)
    - For isalpha: if(isalpha(ch)) // if ch is a letter, isalpha returns true otherwise returns false
- Read the next character from the input file
- After the loop exits, print out the information (number of vowels/consonants, number of lines, average of number of vowels (or consonants) per line as shown in the sample solution. **Average value is printed out to 3 decimal places of precision**