Pair Programming 6 Activities

* **If these instructions make NO SENSE to you, then you need to stop right now and read Chapter 6 of your textbook and ALSO listen to the video lecture for Week 8.**
* **Always use the pair programming tests to ensure your program works properly.**
* **Take a screen shot of each execution in the tests.**
* **Download the source code file for inclusion in the turn in document.**
* **Turn in pair programming activities using the pair programming turn in document.**
* **It is each individual’s responsibility to turn in the assignment and pair programming is graded individually so make sure you share the work you and your partner did together as you go.**
* **Do not share work with your partner that you did not do together.**
* **Make sure you have your partner’s name, username and contact information such as Pellissippi WebMail.**

6a. (4 points) On ps11.pstcc.edu, copy ~caarnold/cisp1010/chapter6.cpp to your pair programming directory and rename it to pp6a.cpp (**cp ~caarnold/cisp1010/chapter6.cpp pp6a.cpp**). Create a file named ***pp6ainput.dat*** and type one integer in it, 55. Compile and execute ***pp6a.cpp***. Check to see if you now have a new filein your current directory named ***pp6aoutput.dat*** that has the words “The number is: 55”. Add code to ***pp6a.cpp*** to do the following:

* Read multiple integers from input.dat and write them to output.dat until the end of the input.dat file. Edit input.dat and put several integers in it. Execute this program and take a screen shot of output.dat.
* Ask the user for the name of the input file and check to see if it exists. If it does not, print the error message “File xxxx does not exist” where xxxx is the name of the file. If it does exist, ask the user for the name of the output file, read numbers from the input file and write them to the output file until the end of the input file. Do not ask the user for the name of the output file if the input file does not exist. Follow the test instructions and take a screen shot of this code executing.
* Check to make sure the input file isn’t empty. If it is empty, print the error message “File xxxx is empty.” where xxxx is the name of the file. If it is non-empty, ask the user for the name of the output file, read numbers from the input file and write them to the output file until the end of the input file. Do not ask the user for the name of the output file if the input file is empty. Follow the test instructions and take a screen shot of this code executing.
* Write the output as dollar amounts by writing two small functions. The first reads the number from the input file. It has two parameters, the *ifstream* and the number. The post-condition is that the actual parameters are changed. The second writes the number to the output file formatted as a dollar amount, such as $123.45, with a $ and exactly 2 digits to the right of the decimal. It also has two parameters, an *ofstream*, whose actual parameter is changed in the course of writing, and the number to be written. Follow the test instructions and take a screen shot of the code executing AND of the contents of output.dat after the program executes.

6b.(2 points) Write a C++ program in a file called *pp6b.cpp* that opens a file named *pp6b.dat* that has the following data in it (you’ll have to create pp6b.dat)

AX013 1.0

BX123456 1234.56

ABNB9876152345 99999.99

The data are account numbers and balances. Account numbers are, at most 14 characters. Balances are, at most, 8 characters (no more than 99999.99). Using a loop until the end of file, read an account number and balance then write these to standard output in the following format shown below. Use the I/O manipulators setw, left, and right. Set the precision/fixed/showpoint so that all numeric data is written with exactly two digits to the right of the decimal.

Account Number Balance

-------------------------

AX013 $ 1.00

BX123456 $ 1234.56

ABNB9876152345 $ 99999.99

6c.(2 points)In a file named *pp6c.cpp*, write a ***function*** *named* ***readline*** that uses a *do while loop* to read all characters from standard input, including whitespace characters, until the function reads the newline character, ‘\n’. Then, write a main function that prompts/reads one word from the user, calls the readline function to read past all the rest of the characters the user might have typed, then prompts/reads another word from the user. Print out both words.

6d. (2 points) In a file named **pp6d.cpp**, write a program that asks the user for one character using the get function. Then test the character and print a message if that character is: 1) upper case, 2) lower case, 3) a digit, 4) white space and/or 5) is an alphabetic character. Use the functions listed in Appendix 4 of your textbook, such as *isdigit, etc.* You will have to include the <cctype> header file.