**COP 2334: C++ Programming**

**Programming Project #4**

**Project Outcomes:**

Develop a C++ application that:

* Uses C++ functions for procedural abstraction and Functional Decomposition
* Utilizes file input and output to perform computing tasks including generating numerical data from numerical input

**Prep Readings:**

Problem Solving with C++, Chapters 5 and 6.

**Project Requirements:**

1. You should do a functional decomposition to identify the functions that you will use in the program. These should include reading and writing the files, tallying and showing statistics, etc.

2. The program should prompt the user for the output file name only. Obviously, this should be taken care of in a function.

3. Sales Records are in a text file named **TheSales.txt**. You MUST use this file name for the input.

a. The first entry will be an integer that tells how many sales people there are.

b. Following that will be an integer that tells you how many weeks of data are in the file.

c. Following that integer will be a sales person's first name, middle initial, and last name,

d. Followed by the daily sales for that employee for the proper number of weeks.

e. Each week will have FIVE days of sales data as doubles. So, a file for 3 sales people and 2 weeks might look like this:

*3*

*2*

*firstName1 A lastName1*

*20.00 25.00 30.90 40.00 55.50*

*20.00 25.00 30.90 40.00 55.50*

*firstname2 B lastName2*

*30.00 24.00 45.00 67.00 65.50*

*56.90 87.00 43.50 56.98 55.40*

*firstName3 C lastName3*

*62.00 34.50 12.50 34.00 34.90*

*70.00 80.00 90.00 65.00 39.00*

4. The program must be able to handle however many salespeople are specified (1 to 10 salespeople) and however many weeks of data are specified (1 to 10 weeks). The data must be organized exactly as specified above. Be sure to read the file that way.

5. The output will go to an output file name that you get from the user, with a format that you must design yourself. It’s recommended that you determine what the output file format on paper or in a text editor to serve as a reference when writing the file output code. The file must contain the following information, all of which must be properly labeled so that it is easy to know what the numbers refer to:

*a.* The number of sales people that were processed

*For the sample input above, 3 sales people were processed*

b. The number of weeks of sales that were processed

*For the sample input above, 6 weeks of sales data were processed*

c. The last name only of each sales person followed by total and average-sales-per-day for each week

*For the sample input above, salesperson “lastName1” had a total sales for week 1 of $171.40 and an average sales per day of $34.28*. *This salesperson had the same result for week 2.*

d. Subtotal sales and average sales-per-week for each sales person over all weeks

*For the sample input above, salesperson “lastName1” had a subtotal sales (all weeks) of $342.80 and average sales-per-week of $171.40*

e. Grand total sales and average-sales-per-week for all salespersons over all weeks

*For the sample input above, the grand total of all sales for all weeks is $1395.98 and the average sales-per-week for all salespersons is $697.99*

6. If the input file or output file opening fails, print an error message and exit.

**Implementation Notes:**

1. You will need to declare several variables. Try to use the least number of variables possible. You should consider the minimum number of running totals that you should keep as you read through the sales data in the input file. For some information (e.g names), you may not need to hold onto it for the entire duration of the program, which means you may be able to reuse some variables multiple times. You should not need to use arrays in this program.
2. You may assume that all weeks will have 5 days of sales reported.
3. You can assume the correct number of names and middle initials will be there and the correct number of sales figures.
4. **Note that you can test your program on data that is easy to hand tally so that you know the correct output you SHOULD get before your program ever runs. That is the ONLY way to know that your program works properly.**
5. **BE SURE to use the proper names for the output file and to use the proper input file format. You will lose points if you do not.**

**Important Notes:**

Compile and run your program one last time before submitting it. Follow the instructor Programming Project Submission Instruction posted on elearning.

Projects will be graded on whether students correctly solve the problem, and whether they adhere to good programming practices including use of comments as specified in the materials, using meaningful variable names, proper indentations, and use of functions to decompose the problem.

Projects must be submitted by the time specified as the due date. No late project will be accepted

Please review UWF's academic conduct policy. Note that viewing another student's solution, whether in whole or in part, is considered academic misconduct. Also note that submitting code obtained through the Internet or other sources, whether in whole or in part, is considered academic misconduct. All programs submitted will be reviewed for evidence of academic misconduct, and all violations will be handled accordingly.