Final Programming Project Assessment (400 Points toward Course Grade)

Instructions: The following programming problem can be solved by a program that uses three basic tasks-Input Data, Process Data, and Output Results. To process the data, use file, looping, array, decision, accumulating, counting, find min/max and sorting techniques. First, create an MS Word document containing a **hierarchy chart** and a **data flow diagram** to organize your program modules. Second, create a **pseudocode** program using Notepad++ to solve this problem. Third, create the program with **RAPTOR**. You MUST use Modular Programming techniques by using Sub Modules (Sub Charts in RAPTOR) in your program. Your "main" module should not be very large.

Problem Statement

Ledger's Furniture Store has 10 salespeople. Ledger's wants to produce a combined monthly sales report for all salespeople. Ledger's wants you to write a program that will allow them to enter data in any order, save the unsorted data to a file, and then produce a file in alphabetical order by last name of the salespeople. The unsorted output file should be named "sales_unsorted.txt" and the sorted file should be named "sales_sorted.txt".

Your *unsorted* file output should include:

- A list of the salespeople’s names and the monthly sales for each of them.

Your *sorted* file output should include:

- A list of the salespeople’s name and the monthly sales for each of them.
- At the bottom of the report, list the following:
    - The total combined sales for all salespeople.
    - The average sales for all salespeople.
    - The salesperson with the lowest sales for the month.
    - The salesperson with the highest sales for the month.

You must submit the unsorted and sorted files with your RAPTOR program to show that your program ran correctly.

Here is your *sample test data* to use with your program. This is to be typed in as user input.

Salesperson	Monthly Sales
Joan	$ 1525
Bob	$ 1935
Lisa	$ 2550
Tina	$ 1745
Corey	$ 3025
Chummily	$ 1420
Rick	$ 4560
Nancy	$ 2645
Holly	$ 1489
Frank	$ 4450

Do not worry about the formatting of your output files since RAPTOR does not support formatting controls.

The *unsorted* file should look similar to this:

```
Joan     Sales: $ 1525
Bob     Sales: $ 1935
Lisa     Sales: $ 2550
Tina     Sales: $ 1745
Corey     Sales: $ 3025
Chummily Sales: $ 1420
Rick     Sales: $ 4560
Nancy     Sales: $ 2645
Holly     Sales: $ 1489
Frank     Sales: $ 4450
```

The *sorted* file should look similar to this:

```
Bob     Sales: $ 1935
Chummily Sales: $ 1420
Corey     Sales: $ 3025
Frank     Sales: $ 4450
Holly     Sales: $ 1489
Joan     Sales: $ 1525
Lisa     Sales: $ 2550
Nancy     Sales: $ 2645
Rick     Sales: $ 4560
Tina     Sales: $ 1745
=====================

Total Sales: $ XXXXX

Average Sales: $ XXXX
Salesperson with Lowest Sales: name
Salesperson with Highest Sales: name
```

Other Requirements:

- Documentation: Use the "Comments" feature to document each symbol in the flowchart. You do this by right-clicking the symbol and selecting "Comment." Be sure to identify the data type of each variable used in your comments. Be sure to explain what each formula and function does. Be sure to explain what each of the other symbols in the flowchart does in a comment.
- Test and debug your Program: Use the sample input data, run the program, then *check your answers* with a calculator or Excel. If something did not match up, then fix your program.
- Program must execute and produce correct output.
- Read this page again to be sure you covered all requirements.
- See the Programming Project Rubric for grading principles.
- Extra Credit: Use Object-Oriented Programming Techniques learned in chapter 11.

Submission Instructions:

- **You will submit 5 files for this project**. You must submit the **Hierarchy Chart** and **Data Flow Diagram** in a MS Word or PDF file (both in same document), a **Pseudocode Program** in a Notepad++ file, and a Flowchart (from **RAPTOR**) file. Your RAPTOR file will be the **.rap** file created when you save your project.
- Name the RAPTOR file (replacing *LastName* and *FirstInitial* with YOUR name): *LastName_FirstInitial*_Program1.rap (example: **Smith_J_Program_3.rap**).
- You must also submit the unsorted and sorted files (**sales_unsorted.txt** and **sales_sorted.txt**) to show that your program ran correctly.
- Attach your files individually (**no** zip files) to your assignment submission upload. If you find that you made an error and want to resubmit before the due date, you may do so. However, only the LAST files uploaded by the due date will be graded.

Programming Project Rubric

	Unsatisfactory	**Satisfactory**	**Good**	**Excellent**
Delivery	• Completed less than 70% of the requirements. • Not delivered on time or not in correct format (disk, email, etc.)	• Completed between 70-80% of the requirements. • Delivered on time, and in correct format (disk, email, etc.)	• Completed between 80-90% of the requirements. • Delivered on time, and in correct format (disk, email, etc.)	• Completed between 90-100% of the requirements. • Delivered on time, and in correct format (disk, email, etc.)
Coding Standards	• No name, date, or assignment title included • Poor use of white space (indentation, blank lines). • Disorganized and messy • Poor use of variables (many global variables, ambiguous naming).	• Includes name, date, and assignment title. • White space makes program fairly easy to read. • Organized work. • Good use of variables (few global variables, unambiguous naming).	• Includes name, date, and assignment title. • Good use of white space. • Organized work. • Good use of variables (no global variables, unambiguous naming)	• Includes name, date, and assignment title. • Excellent use of white space. • Creatively organized work. • Excellent use of variables (no global variables, unambiguous naming).
Documentation	• No documentation included.	• Basic documentation has been completed including descriptions of all variables. • Purpose is noted for each function.	• Clearly documented including descriptions of all variables. • Specific purpose is noted for each function and control structure.	• Clearly and effectively documented including descriptions of all variables. • Specific purpose is noted for each function, control structure, input requirements, and output results.
Runtime	• Does not execute due to errors. • User prompts are misleading or non-existent. • No testing has been completed.	• Executes without errors. • User prompts contain little information, poor design. • Some testing has been completed.	• Executes without errors. • User prompts are understandable, minimum use of symbols or spacing in output. • Thorough testing has been completed	• Executes without errors excellent user prompts, good use of symbols, spacing in output. • Thorough and organized testing has been completed and output from test cases is included.
Efficiency	• A difficult and inefficient solution.	• A logical solution that is easy to follow but it is not the most efficient.	• Solution is efficient and easy to follow (i.e. no confusing tricks).	• Solution is efficient, easy to understand, and maintain.