Project #6 – CO_2 Data Entry & Stats

1 Problem Overview

Assume you are participating in the iRISE effort and have checked out a CO_2 meter. You want to write a program to help you enter batches of data pertaining to specific locations over time, for example readings from the bus every day for a week.

2 User Interface

2.1 Initial Input

Ask the user for their name and the location of the readings. Here is an example of what that conversation might look like:

```
Enter name: Bill Barry
Enter location: Metro Bus 514
```

Validate each input before moving on, ensuring that the string is not empty, i.e., that the user doesn't just press <Enter> instead of entering valid data. Keep pestering the user until they enter valid data, regardless of how many tries it takes.

2.2 Reading Input

Now ask the user for the CO_2 reading data. Ask for month, day, year, and CO2 level. Keep asking them for readings until they enter 0 for the month; then, without asking for further data, move on with he program. Here is an example of a batch of 3 readings:

```
Enter month (or 0 to exit): 6
Enter day: 26
Enter year: 2022
Enter CO2 reading: 865

Enter month (or 0 to exit): 6
Enter day: 27
Enter year: 2022
Enter CO2 reading: 937

Enter month (or 0 to exit): 6
Enter day: 28
Enter year: 2022
Enter CO2 reading: 1006

Enter month (or 0 to exit): 0
```

Validate each input before moving on, ensuring that the month is in the range 0-12, day 1-31, year 2022-2099, and CO_2 level 1 to 10000. Keep pestering the user until they enter valid data, regardless of how many tries it takes.

2.3 Output

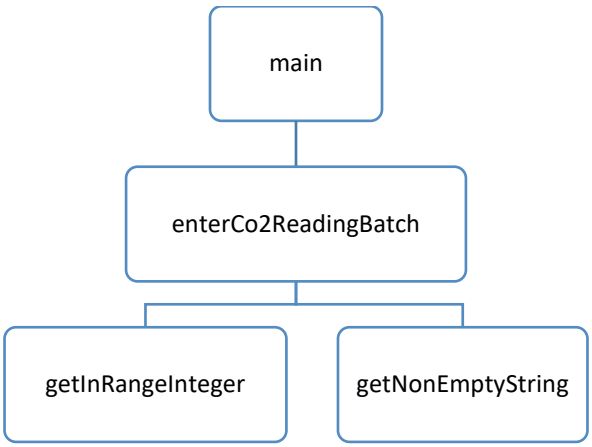
Once the batch entry is complete, show a report that gives the statistics on the batch. The report should look very much like this one, depending on data entered. For the sample data shown above, the report would look like this. Note that average is displayed rounded to the nearest integer.

```
------------------------------------------------
Recorder name:       Bill Barry
Reading location:    Metro Bus 514

Number of readings:       3
Minimum CO2 Level:      865
Maximum CO2 Level:     1006
Average CO2 level:      936
------------------------------------------------
```

3 Functions

3.1 Call Hierarchy



3.2 Function Specifics

Function Name	Parameters	Returns	Description
main	none	none	Calls enterCo2ReadingBatch, receives its returned data, and prints the report
enterCo2ReadingBatch	none	name, location, minimum level, maximum level, reading count, and average level	Performs data entry, returns statistics
getInRangeInteger	min value allowed, max value allowed, prompt[1]	valid integer	Asks the user for data, validates it (keeps pestering the user until they comply), and returns an integer guaranteed to be in the specified range
getNonEmptyString	prompt	valid string	Asks the user for data, validates it (keeps pestering the user until they comply), and returns a non-empty string

[1] Prompt is a string giving context to the input, e.g., “month” when entering month data.

4 Code Specifications

- At the **bottom** of the program, you should have a call to main().
- Include header comments at the top of each file. Include your name, the date, and a brief description of what the program does.
- Include comments for each section saying what's going on in the lines of code below.
- Use comments elsewhere as you think they help guide the reader. Don't overdo, though. Not every line needs a comment; think about describing a section of related code.
- Use blank lines to separate sections and provide visual "breathing room."
- Use descriptive variable names.
- For formatting, use the format *function* only, not the format *method*, "f strings," etc.
- Do not use lists or other tech we haven't covered; what has been presented in class is sufficient.

5 Hints

- Every function except main will have one loop.
- Use *exactly* the call hierarchy. Helper functions are allowed but are probably unnecessary here.

6 Testing

- Develop an appropriate number of test cases.
- Document your testing and results in comments at the bottom of the program as shown below.

7 Summary

At the bottom of your program, add comments that answer these questions:

- How did you approach this assignment? Where did you get stuck, and how did you get unstuck?
- How did you test your program? What doesn't work as you'd like, perhaps things that you'd like to fix as you learn more?
- What did you learn from this assignment? What will you do differently on the next project?

8 Grading Matrix

Area	**Percent**
Data entry and looping	20
Integer validation	20
String validation	15
Correctness of calculations	5
Output content	10
Output formatting	10
Test cases	10
Internal documentation	5
Summary report	5
Total	**100**