HW03 Implement a Solution

Purpose: To demonstrate ability to implement a modularized quality code solution using c concepts learned in class so far.
Effort: Individual CS Academic Integrity .pdf

- Reminder if you share code/solutions or use someone else's code you will receive a 0 and be reported to the CS chair and UCCS Dean of Students

Points: 100
Read Rubric in canvas
Due date: Look in canvas
Deliverables: Reflection as doc or pdf and lastnameFirstNameHW03.c file. Submit files separately and **not in a zip file.**

Requirements: The parking garage owner wants a daily summary of the total number of hours cars were parked and total amount collected for the day.
The parking garage charges a $3.00 minimum flat rate fee to park for up to and including 3 hours and an additional $0.75 charge per hour for each hour or part thereof over 3 hours. Partial hours can be entered. The maximum number of hours a car can be parked is 24 hours and the maximum charge is $12.00.

Acceptance criteria:

- If the number of hours input is not a valid numeric value greater than 0 and less than or equal to 24 report an error and repeat the question. For example:

```
Enter the number of hours the car was parked or enter -1 to quit.
d
You did not enter a number
Enter the number of hours the car was parked or enter -1 to quit.
```

- When valid data is inputted the parking fee should be calculated and the car number, hours and charge should be displayed. For example the following you can accept as a valid value since scanf will read the 3.4. The charge is based on any part over 3 hours and will use 4 hours to determine the charge.

```
Enter the number of hours the car was parked or enter -1 to quit.
3.4f
 Car        Hours       Charge
  1          3.4         $ 3.75
```

```
Enter the number of hours the car was parked or enter -1 to quit.
.5
 Car        Hours       Charge
 2          .5          $ 3
```

```
Enter the number of hours the car was parked or enter -1 to quit.
19.2
 Car        Hours       Charge
```

```
3           19.2          $12.00
```

- The user can continue entering hours until -1 is entered. When -1 is entered the total number of cars, total number of hours and total charges collected should be displayed. For example:

```
Enter the number of hours the car was parked or enter -1 to quit.
-1

Parking Garage Summary
Total Cars          Total Hours          Total Charge
 3                   23.1                $18.75
```

- If there were no cars parked then display no cars parked. For example:

```
Enter the number of hours the car was parked or enter -1 to quit.
-1

Parking Garage Summary
There were no cars parked today.
```

Implement Quality Maintainable Modular Solution

Resources: See SDLC Implement and Test slides 16 to 26, What's Needed to Get Your Code Quality Match ISO Standard 25010 and Coding Mindset

Description: Implement a solution that is

1. Solution Functionally Complete and Correct
2. Quality Readable Code
3. Maintainable and Secure
4. Do not implement concepts not covered yet in class.

Your solution should have at least 3 (can have more) user defined functions besides main:

- Get valid input
- Calculate charge
- Display Parking Totals Summary

Implement a solution that does not hard code the values and uses the following constants

- minimum hours at flat rate: MIN_HOURS_AT_FLAT_RATE
- minimum flat rate: MIN_FLAT_RATE_CHARGE
- Additional charge for every hour over minimum hours: ADDITIONAL_HOURS_RATE
- maximum hours allowed: MAX_HOURS_ALLOWED
- Maximum charge: MAX_CHARGE

Meaning we will change the global constants to test the program. For example, what if the customer changes their pricing like this

The parking garage charges a $**2.50** minimum flat rate fee to park for up to and including **2** hours and an additional $**1.25** charge per hour for each hour or part thereof over **2** hours. Partial

hours can be entered. The maximum number of hours a car can be parked is **24** hours and the maximum charge is $**15.00**.

Learnings and Reflection (In your own words)

Explain:

verb (used with object)

1. to make plain or clear; render understandable or intelligible:
   to explain an obscure point.
2. to make known in detail:
   to explain how to do something.

1. What are you proud of from this assignment?
I am proud of how thorough my pseudocode and test cases ultimately were, resulting in each and every function working perfectly the first time.

2. Explain the secure coding practices you implemented. Give examples
   1. When counting cars in main, I used an unsigned int to track car numbers and matching arguments to pass that number to the print functions.
   2. When printing out car data in the print functions, I used the unsigned integer format specifier(%u) rather than the integer specifier(%d).
   3. I collected the return value of scanf() and used it to validate the input in getValidInput().
   4. All print statements throughout the program were either puts() or two argument printf() statements: no single argument printf() statements were used.
   5. The input buffer was cleared each time an input was gathered in getValidInput().
   6. I validated numeric inputs using a range defined by flexible constants.

3. From What's Needed to Get Your Code Quality Match ISO Standard 25010 - Data Analytics and Coding Mindset explain and give examples. **Include at least one example from your design or your code for each explanation.**

3.1 Explain what you did to achieve functional completeness..
To achieve functional completeness, I poured most of my mental resources into the pseudocode and design stage. First, I made certain that I had all the functions I needed to make a maintainable, human readable program. Second I double checked all of the functions, main being checked last, for their own, individual functional completeness.

3.2 Explain what you did to achieve functional correctness.
Functional correctness was achieved by again focusing more time and effort on the design stage. I made sure to make and double check test cases for each function. Then I moved on to the code, and only implemented one function at a time, running test cases using the design document as a guide. The test cases were deleted from the final code, but can still be viewed on my github on the file's history page: https://github.com/ParadigmMango/CS2060ClassCode/commits/main/examples/homework/HW03/hw03.c.

3.3 Explain what you did to achieve quality readable code.

Again, the design stage was the crucial step for the outcome of the code. I made sure that the pseudocode itself was well readable (i.e. concise variable names, good structure, etc.), then after I implemented the code I commented on it extensively.

3.4 Explain what you did to make your code maintainable and extendable

1. I designed the pseudocode to split the program into as many useful functions as reasonably possible and create modularity.
2. I used constants instead of hard coding data.
3. I invested in making the code readable, as seen above.
4. I used doxygen style comments which are themselves verbose, but can also be generated into documentation using doxygen. While not included in the assignment submission, the generated html documentation can be downloaded here: https://github.com/ParadigmMango/CS2060ClassCode/tree/main/examples/homework/HW03/html.
5. I created many test cases to achieve testability.

3.5 Explain what secure coding standards you followed to validate input

1. I collected the return value of scanf() to determine if the input is numeric.
2. I used a range to determine whether the numeric input was valid.

4.How did you approach this problem to implement an easily modifiable solution to handle the possible customer changes for the pricing and hours?

I poured my heart and soul into the design document to ensure the final code would meet the assignment's standards. In order to create flexibility for customer pricing and hours, I did two things during that design phase:

1. I created constants to allow for easy changes if the values driving the calculations needed to be changed.
2. I created a separate calculations function in case the logic behind the calculations needed to change.