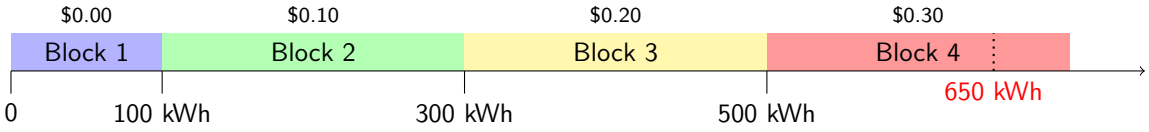
Programming Assignment 2

Filename: `pa2_surname.c`

In a fictional city of Irston, households are powered by Irston Regional Supply (IRS Power), which charges for electricity using an inclining block rate system. This means that electricity is not billed at a single flat rate. Instead, the total electricity usage is divided into ranges (blocks). Each block is charged at its own rate. The billing structure is as follows:

Block	Usage Range (kWh)	Rate per kWh
1	0 – 100	\$0.00
2	101 – 300	\$0.10
3	301 – 500	\$0.20
4	501 and above	\$0.30

Only the portion of electricity usage that falls within a given block is charged at that block's rate. Lower blocks are always filled first before higher blocks are applied.



IRS Power has asked you to write a C program that computes the bill for a household given their usage in kWh. The usage values will be entered as whole numbers. This computed amount will be referred to as the base bill. If it exceeds \$100, add a 10% surcharge and include it in the final bill. Store the monetary values in variables of type `double`. All monetary amounts must be displayed with exactly two decimal places using `printf()`.

Example: A Household with Usage of 650 kWh

Step 1: Apply Block Rates

$$\begin{aligned}
\text{Block 1 (first 100 kWh @ \$0.00)} &: 100 \times 0.00 = \ \ 0.00 \\
\text{Block 2 (next 200 kWh @ \$0.10)} &: 200 \times 0.10 = 20.00 \\
\text{Block 3 (next 200 kWh @ \$0.20)} &: 200 \times 0.20 = 40.00 \\
\text{Block 4 (remaining 150 kWh @ \$0.30)} &: 150 \times 0.30 = 45.00
\end{aligned}$$

$$\text{Base Bill} = 0.00 + 20.00 + 40.00 + 45.00 = \mathbf{105.00}$$

Step 2: Apply Surcharge Since the base bill exceeds \$100, a 10% surcharge is applied:

$$10\% \text{ of } 105.00 = \mathbf{10.50}$$

Step 3: Compute Final Bill

$$\text{Final Bill} = 105.00 + 10.50 = \mathbf{115.50}$$

Three sample runs are provided below. Anything that is underlined represents user input. If the user provides an invalid usage value (e.g., a negative number), the program should display **`Invalid Input`** and terminate immediately. You may assume that the usage value will fit within the `int` data type.

Important You are not allowed to use the `exit()` function.

Sample Run 1

```
Enter usage (kWh): -75
Invalid Input
```

Sample Run 2

```
Enter usage (kWh): 250
Base bill: $15.00
Final bill: $15.00
```

Sample Run 3

```
Enter usage (kWh): 650
Base bill: $105.00
Surcharge (10%): $10.50
Final bill: $115.50
```

Grading Rubric

Criteria	0	5	10
Compiling the Source Code on the Eustis Server *The solution developed incorporated only concepts covered in class (except those explicitly allowed in the instructions).*	The source code could not be compiled on the Eustis server. Therefore, the entire submission is not graded anymore and will be marked as 0/100. **OR** The solution included code/s OR function/s not covered in class. Therefore, the entire submission is not graded anymore and will be marked as 0/100.	*(Not Applicable)*	The source code can be compiled on the Eustis server. **AND** The solution included code/s AND function/s that were covered in class.
Passed Sample Test Cases *For each of the sample runs provided. Max of 10 per test case.*	There is no implementation at all.	The program logic is incorrect; however, some effort was exerted (including cases where the program does not produce any output). **OR** The program produced partially correct output (e.g., incorrect prompt used; wrong spelling; missing characters/symbols; missing/exceeding whitespaces; or using the wrong case for letters).	The program passed all the sample runs provided where the output **matched exactly** the expected output.
Passed Secret Test Cases *The number of secret test cases will not be disclosed. However, they will be weighed such that the overall sum is 50%. Each secret test case will be binarily graded as either correct or not. Max of 50 as the sum for all test cases.*	*(Binary and Weighted; Refer to Description)*	*(Binary and Weighted; Refer to Description)*	*(Binary and Weighted; Refer to Description)*
Code Readability *Refer to the style guide here.*	The code was not very readable **AND** lacked meaningful comments.	Either the source code was not very readable because of poor indentation **OR** there was a lack of meaningful comments throughout the source code.	The source code was readable because proper indentation was followed. Further, relevant and meaningful comments were incorporated throughout the source code.