IT 140 Module Four Assignment Guidelines and Rubric

Overview

In this assignment, you will gain more practice with designing a program. Specifically, you will create pseudocode for a higher/lower game. This will give you practice designing a more complex program and allow you to see more of the benefits that designing before coding can offer. The higher/lower game will combine different programming constructs that you have been learning about, such as input and output, decision branching, and a loop.

Higher/Lower Game Description

Your friend Maria has come to you and said that she has been playing the higher/lower game with her three-year-old daughter Bella. Maria tells Bella that she is thinking of a number between 1 and 10, and then Bella tries to guess the number. When Bella guesses a number, Maria tells her whether the number she is thinking of is higher or lower or if Bella guessed it. The game continues until Bella guesses the right number. As much as Maria likes playing the game with Bella, Bella is very excited to play the game *all* the time. Maria thought it would be great if you could create a program that allows Bella to play the game as much as she wants.

Prompt

For this assignment, you will be *designing* pseudocode for a higher/lower game program. The higher/lower game program uses similar constructs to the game you will design and develop in Projects One and Two.

1. Review the Higher/Lower Game Sample Output PDF for more detailed examples of this game. As you read, consider the following questions:
    - What are the different steps needed in this program? How can you break them down in a way that a computer can understand?
    - What information would you need from the user at each point (inputs)? What information would you output to the user at each point?
    - When might it be a good idea to use “IF” and “IF ELSE” statements?
    - When might it be a good idea to use loops?

2. **Create pseudocode that logically outlines each step of the game program so that it meets the following functionality:**
    - Prompts the user to **input** the lower bound and upper bound. Include input validation to ensure that the lower bound is less than the upper bound.
    - Generates a random number between the lower and upper bounds
    - Prompts the user to **input** a guess between the lower and upper bounds. Include input validation to ensure that the user only enters values between the lower and upper bound.
    - Prints an **output** statement based on the guessed number. Be sure to account for each of the following situations through the use of **decision branching**:
        - What should the computer output if the user guesses a number that is too low?
        - What should the computer output if the user guesses a number that is too high?
        - What should the computer output if the user guesses the right number?
    - **Loops** so that the game continues prompting the user for a new number until the user guesses the correct number.

3. **OPTIONAL:** If you would like to practice turning your designs into code, check out the optional 9.1 LAB: Higher/Lower Game in zyBooks. This step is optional, but will give you additional practice turning designs into code, which will support your work in moving from Project One to Project Two.

What to Submit

Submit your completed pseudocode as a Word document of approximately 1 to 2 pages in length.

Module Four Assignment Rubric

Criteria	Exemplary	Proficient	Needs Improvement	Not Evident	Value
Pseudocode: Logical Steps	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or efficient manner (100%)	Creates pseudocode that logically outlines each step of the program so that it meets the required functionality (85%)	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include creating more pseudocode for all the functionality included in the program (55%)	Does not attempt criterion (0%)	35
Pseudocode: Input/Output	N/A	Determines user inputs and outputs based on the given scenario (100%)	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include accounting for all inputs and outputs in the design document (55%)	Does not attempt criterion (0%)	30
Pseudocode: Program Flow	Exceeds proficiency in an exceptionally clear, insightful, sophisticated, or efficient manner (100%)	Uses decision branching and loops to control program flow (85%)	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include accounting for all the paths the user can take through the program (55%)	Does not attempt criterion (0%)	35
				Total:	100%