Programming HW #8 – 100 Pts

Use 32-bit MASM x86 Assembly Language to implement a game of Hangman

a. In the data section of your code define at least ten words, with at least six letters each, as strings.
b. Have the program randomly generate an integer from 1 to 10 (or however many words you stored) to correspond to one of the words. The initial display for the word should be a series of underscores.
c. Prompt the user to guess a letter.
   1. If the user guesses correctly and the word is still not completely solved, then prompt the user for another letter. Display the correctly guessed letter in the appropriate position(s) in the word, leaving the other positions as underscores.
   2. If the user guesses incorrectly, then add the next body part to the hangman. If the hangman is not yet complete, prompt the user for another guess. If the hangman is complete, inform the user that the game is over, and he or she has lost. Display a list of incorrectly guessed letters.
   3. If the user has successfully guessed all the letters, display a winning message.
d. The hangman body parts are as follows: Head, Body, Left Arm, Right Arm, Left Leg, and Right Leg. You can just list the body parts, as opposed to drawing the hangman on the output display

   Up to 25 points extra credit is available for an accurate hangman output display throughout program execution. The amount of extra credit points to be awarded as the discretion of the instructor and is dependent on quality.

e. Letter entries should not be case sensitive.
f. Whether the user wins or loses, include an option to play again that is not case-sensitive.

Do not use magic numbers. Name and initialize variables in the .data section of your programs.

Submit your **source code (.asm)** files to the assignment drop-box by the due date. **Submit a single .asm file.** Do not submit your .sln project file. Do not submit Word or PDF files containing your code.

To ensure consistent grading, please include all necessary INCLUDE and INCLUDELIB statements directly within your code, exactly as directed in the course Syllabus, Policy Contract, and recent announcement. I copy and paste submissions into a standard Visual Studio project for evaluation. Due to the volume of submissions, I'm unable to modify code to accommodate individual Visual Studio configurations. This requirement also helps maintain academic integrity by ensuring only approved files and libraries are used.

Build and debug your code using Visual Studio 2022.

Comment Requirements:
There must be a preamble block of comments at the beginning of the program that includes at least the following information: Your name, course number/section/title, program title, and date

There must be a block of comments before the main procedure of your program, as well as before each additional procedure (if applicable) that describes the inputs, outputs, memory usage, register usage, and functional description of the procedure.

The program must also contain **non-syntax based individual line comments for EACH line** that make the functional goals and processes of the program self-documenting. These comments should be placed in line with each instruction (not above or below). Use tabs to line things up!

Grading:

Functionality: 35 Pts
Output Formatting: 15 Pts
Program Structure and Coding Concepts: 25 Pts
Comments and Internal Documentation: 25 Pts