You will create a word guessing game that allows the player to guess the letters of a secret word. At the start of the game, the player is shown only how many letters the word contains through a set of dashes. When a letter matching one in the word is guessed, it replaces the appropriate dash. The game continues until the entire word is guessed letter by letter or when the player chooses to guess the entire word.

**Specifications**

For this assignment, you will need to use conditionals, looping logic, string methods and error checking logic. Guesses will not be case sensitive.

The word guess game, called WordGuess, is played between the computer and a single player. The secret word is BRAIN. At the start of the game, 5 dashes are displayed (- - - - -), one for each letter of the word. The player is repeatedly prompted for a letter guess. When a letter matching one in the word is guessed, the letter replaces the corresponding dash. Letters may be entered as upper or lower case. However, only uppercase letters should be displayed. If the user enters an exclamation point, (!), the player is prompted to guess the word. At that point the player either wins (a correct guess) or loses (an incorrect guess). Alternatively, the player can continue to guess letters until the entire word is revealed. The game ends by showing the player the total number of guesses which were made.

**Sample Screen Layout**

Your screen must look like this

**Welcome to the WordGuess game!**

**Your word: \_ \_ \_ \_ \_**

**Enter a letter (! To guess the entire word): A**

**Your word: \_ \_ A \_ \_**

**Enter a letter (! To guess the entire word): v**

**Your word: \_ \_ A \_ \_**

**Enter a letter (! To guess the entire word): !**

**What is your guess? Brain**

**You won! The secret word is BRAIN. You made 3 guesses.**

**Program Checklist**

|  |  |  |
| --- | --- | --- |
|  | **Description** | **Marks** |
| **Knowledge** | **Programming Concepts:**   * Appropriate declaration of variables with meaningful names and  suitable data types * Appropriate use of object classes for input and output * Appropriate use of arithmetic operators to perform program calculations * Appropriate use of conditional statements, loop logic, string classes, relational operators to evaluate program conditions |  |
| **Thinking** | **Algorithms:**   * IPO Chart, pseudo code, or flowchart provides detailed step-by-step instructions to properly implement the program specifications. * This must be done before coding is started |  |
| **Communication** | **Program Header:**   * contains programmer’s name, course code, date program written, and a comprehensive description of the purpose of the program   **Internal Documentation:**   * comments are used appropriately within the program and provide a meaningful summary of major processes   **Formatting:**   * program source code is properly indented where required and contains appropriate white space for readability * User interface is courteous, esthetically pleasing, and free of spelling and grammar errors |  |
| **Application** | **Implementation:**   * Output matches sample layout. * User input is checked for data type acceptability and user is informed if their entry is not matching the type expected. * Calculations to accumulate guesses are accurate. * Program source code is efficient and executes as required with no syntax or logic errors |  |