Assignment A12. Wordle

Your assignment is to implement the **Wordle** game, that you can try playing online here:
https://www.nytimes.com/games/wordle/index.html

If you are not familiar with Wordle, I suggest you play this to understand how it works.

Rules of Wordle

There is a secret 5-letter word you have to guess.

You type in your guess, and different letters in your guess are colored blue, green or gray:

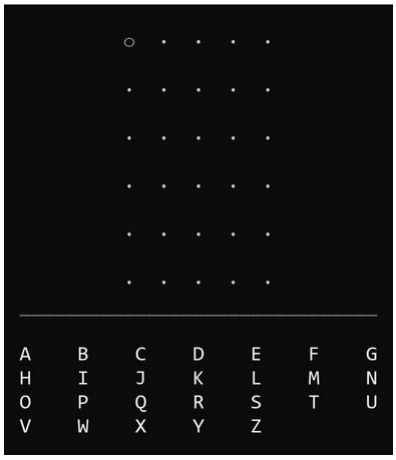
- Gray: the given letter is not in the secret word
- Blue: the letter is found in the word, but not in this position
- Green: The letter is found in the word, and is in the correct position

With this feedback, you can improve your guess. You have 6 tries in which the word must be guessed.

Assignment

Initial Display

Implement this as a console application. The keyboard interface should be very similar to the online wordle game. Initially, you get a 'blank grid' of 6 lines with 5 letters each into which you can enter your guesses. The initial screen should look a bit like this:



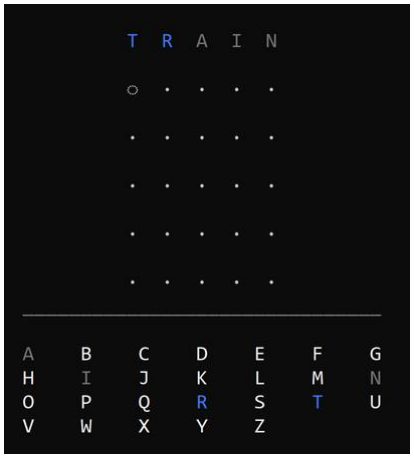
The small circle indicator on the top left shows where the next character you type will go into. The letters at the bottom indicate which letters have been used already, and some hints about whether these used letters are not found in the word, or in the correct / incorrect position etc. The bottom section is just a visual aid that helps solving the puzzle more easily. Your display should look identical to this. Try to center the display *horizontally* in the middle of the console.

Hint: the small dot is the character '\u00b7' and the circle is the character '\u25cc'. You'll need to turn on the Unicode encoding on the console for these to display properly.

Hint: use the Console.WindowWidth property to figure out how many columns the console is displaying, this can help you center the display better.

After typing in a few guesses

Let's assume the secret word is **STORE** and I type in **TRAIN** as my first guess. The display should then look like this:



Note that the **T** and **R** are colored **blue** (which we are going to use to indicate that the letter is in the wrong place compared to the secret word STORE), while **A I N** are colored gray (to indicate they don't occur in the secret word STORE).

The circle cursor is shown on the first column of the second line (to indicate that's where my next typing will go).

Note also the 'keyboard' on the bottom. This has a few letters in gray (to indicate they've been tried, but don't belong), some in blue (to indicate these belong but are in the wrong place) and many in white (to indicate these have not yet been tried).

Hint: Use **Console.CursorLeft** and **Console.CursorTop** to set where the next output will happen, when you use Console.Write

Hint: The flashing cursor will be distracting (since we using the circle symbol to indicate the next position). Use **Console.CursorVisible** to turn it off.

Assume that I type in a few more guesses:

Remembering that the secret word is STORE, you can see how the green letters in lines 2 and 3 indicate the letters in the correct position. You can also see that T is in the wrong position in that line, and so is indicated by blue. On line 4, the letters are still all in white, since I have not yet finished typing in that word and submitted it by pressing ENTER.

Note also the keyboard display oin the bottom that shows the letters R, S, T in green (the status from the last complete word STARK) several used letters in gray and some unused letters in white.

Keyboard interaction

The keyboard interaction for this game is as follows:

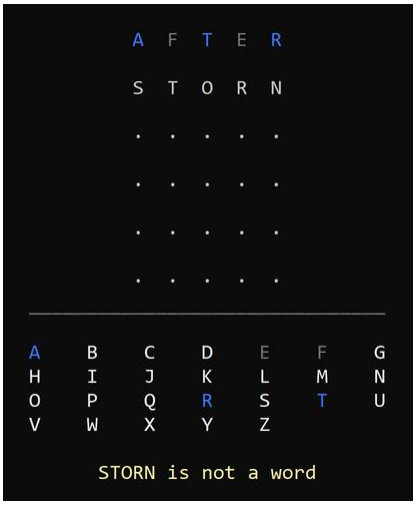
- The circle indicator shows where the next character you type in will go. If you type in a letter from A to Z (or from a .. z), that letter (converted to uppercase) is displayed there, and the circle moves one step to the right (for the next character). If the circle was already in the last column, it disappears (indicating you cannot type in any more letters).
- You can press left-arrow or backspace to 'delete' the last character you typed. The dot indicator reappears there, and the circle indicator moves back one space. If you are already at the first column, the backspace does nothing (in particular, it does not go to the previous word).
- When you have typed in 5 letters like this, you can then press ENTER to submit that word. A word is not automatically submitted when you type in 5 letters; you have to press ENTER. This gives you a chance to go back and correct some of the letters.

- The word you are typing in (even if you have typed in all 5 letters) is displayed in white – the color coding happens only after the word is submitted.
- When you press ENTER, and you have already typed in 5 letters, then that word is submitted (see section below).

Submitting a word

When you type in 5 words and press ENTER, you are trying to *submit* a word. If ENTER is pressed when you have not yet typed in 5 letters, it is ignored. Here is the logic flow for the *submit* action.

1. If the word is not found in the dictionary at all, then the word is not accepted. The game board remains in the same state (so the user can try to delete some characters, and try a different word). A message is shown as indicated:



   At this point, if the user pressed any key, that message disappears and the user can continue attempting a new word. For example, the user could press the back-space key to attempt a new word.
2. The color coding is done for the word that is just typed in: gray / blue / green depending on the letter rules.
3. If the word you just typed in is correct (all green letters would be displayed), the game is over and you have guessed the word correctly (see game over section below)
4. If the word you typed in is incorrect, the circle cursor goes to the next line, and you begin a new guess. There is no way to 'go back' to an earlier guess, so you get a maximum of 6 turns to guess and submit a word. If you press ENTER the sixth time without guessing the word correctly, the game ends with you not having guessed the word.

Game over

The game gets over if you guess the word correctly – the word will then be displayed in green, and a status message like this is displayed below the board:

If you exceed 6 tries and fail to guess the word, the game ends, and the correct word is displayed:

Word lists

I've attached two *word lists* with this. The **puzzle-5.txt** list is a list of 5 letter words that you can use as the secret word. The program should select one of these at random. The **dict-5.txt** is a more extensive dictionary of almost all 5 letter words in the English language. The puzzle-5 list is a subset since there are a lot of more rare or complex words, proper nouns etc in dict-5 that would be difficult to guess.

Thus: you will use the puzzle-5 words for selecting a word initially. But use the dict-5 list to validate if the word the user has typed in is a legal word.

Tips for structuring your application

This is a bit more complex than some of the earlier assignments. The Wordle object has to maintain various bits of internal state. Here are a few of them:

- The secret word that has been selected
- The current row / column position where the user is typing in
- The complete words the user has typed in so far

Try to write a full-fledged DisplayBoard routine that displays the complete state. Each time any bit of state changes, it's simplest to just display the complete board all over again (rather than trying to *incrementally update it*). For this, the CursorLeft, CursorTop properties are useful.

Note: I don't want to see the display scrolling with new updates each time something changes. You should be 'overprinting' the existing display so there is no flicker and it all looks very smooth and seamless.

As a seed, here is the skeleton structure I used to implement this (the screen-shots above are from my implementation). You could use this as an idea to get started:

```
class Wordle {
   // Interface --------------------------------
   // Public interface routine to run the game
   public void Run () {
      ClearScreen ();
      SelectWord ();
      DisplayBoard ();
      while (!GameOver) {
         ConsoleKeyInfo key = ReadKey (true);
         UpdateGameState (key);
         DisplayBoard ();
      }
      PrintResult ();
   }
```