Cheating Hangman

In case you are not familiar with the game Hangman, the rules are as follows:

1. The computer chooses a secret word, and then writes out a number of dashes equal to the word length.

2. The player begins guessing letters. Whenever he guesses a letter contained in the hidden word, the computer reveals each instance of that letter in the word. Otherwise, the guess is wrong.

3. The game ends either when all the letters in the word have been revealed or when the guesser has run out of guesses.

A version of this game has been provided as starter code. The code uses doubly linked lists so that (1) you are exposed to them (2) pointers are reviewed. Avoid using google to find a more complete version. You need to be able to code without copying.

Your assignment is to create a version of the game in which the computer cheats.

The Cheat Method

The trick is this: the computer does not pick a word until it absolutely has to. The game begins with the computer selecting a word length, but not a word.

When you guess a letter (let's say you guess the letter 'N'), the computer consults its word list to see if there are words which don't contain an 'N'. No problem. There are thousands of them! So the computer tells you your guess is wrong. (From this, you can conclude that your first guess will ALWAYS be wrong!). Now let's say you guess the letter 'S'. The computer looks for words which do not contain either an 'N' or an 'S'. Since there are plenty of these, the computer once again tells you your guess is wrong.

Eventually, the computer will not be able to find a word that doesn't contain any of your guesses, so it'll have to pick a word from the dictionary. Even now, the computer can cheat a little bit, because when you make your next guess, the computer will look to see if there is a different word which contains the previous guesses in all the right places, but does not contain the new guess.

You are to design an algorithm which cheats in an efficient way. You would like to avoid looking at all words in the dictionary at every guess. When you reveal letters in the word, the more possible words which match the pattern the better.
Sample Output
To be able to track how well your program cheats, include a count of the number of words that match the displayed pattern. This is an example of the interaction where the number in square brackets is the number of possible words that match. Obviously, the computer is not a very good cheater as once it revealed the i, there were few possibilities left.

You have 8 guesses left...
You’ve used the following letters:
So far the word is
-------- [127142]
Enter your guess: a

Sorry a isn’t in the word
You have 7 guesses left
You’ve used the following letters:a
So far the word is
-------- [127142]
Enter your guess: e

Sorry e isn’t in the word
You have 6 guesses left
You’ve used the following letters:ae
So far the word is
-------- [127142]
Enter your guess: i
That’s right! i is in the word
You have 6 guesses left
You’ve used the following letters:aei
So far the word is
--i-i-- [4]

Enter your guess:o
Sorry 0 isn’t in the word
You have 5 guesses left
You’ve used the following letters:aeio
So far the word is
--i-i-- [4]

Enter your guess:t
Sorry t isn’t in the word
You have 4 guesses left
You’ve used the following letters:aeiot
So far the word is
--i-i-- [3]
Enter your guess: n
That’s right! n is in the word
You have 4 guesses left
You’ve used the following letters: aeiotn
So far the word is
--in-  [2]

Enter your guess: g
That’s right! g is in the word
You have 4 guesses left
You’ve used the following letters: aeiotng
So far the word is
--ing  [1]

Enter your guess:c
That’s right! c is in the word
You have 4 guesses left
You’ve used the following letters: aeotngc
So far the word is
c-ing  [1]

Enter your guess:h
That’s right! h is in the word
You win. The word is chining

**Deliverables**

1. The Visual Studio project containing all of the code. (If you are not using visual studio, submit all of the files needed to run the program.)
2. A description of the algorithm you used to cheat (include in a file named ReadMe file). Point out ways the algorithm could be made to work better.

**Grading**

1. Algorithm cheats effectively and algorithm is described well. (10 points)
2. Correctly identifying the number of words which match the pattern (5 points)
3. Well written and documented code (5 points)