These exercises will be due at the beginning of class on Friday, September 30. Please submit a legible hard copy of your solutions at that time.

0. Ancient texts in languages that used alphabetic writing systems frequently lacked spaces or any other separators between words. Finding the correct way of dividing the stream of letters into words was left as an exercise for the reader.

Design an algorithm for dividing a list of letters into words. You may assume that the file /usr/share/dict/words contains every valid word. If the list can be divided up in more than one way, evaluate alternative division schemes as follows: Consider each word \( w \) that results from the division. If \( w \) is not listed in /usr/share/dict/words, award it 0 points. Otherwise, let \( n \) be the length of \( w \). If \( n \leq 3 \), award \( n \) points; if \( n > 3 \), award \( 2n - 3 \) points. Add the awards for all the words to get the total for the division scheme. The goal of the algorithm is to find the division scheme that produces the highest total.

1. Implement and test your algorithm in Scheme, C, or Java.

2. Propose an algorithmic upper bound on the running time of your algorithm, as a function of the length of the list of characters it is given, and show informally that your implementation satisfies that upper bound.