Student Name: Conrad Chen
UW Campus ID: 9071560800
email: wchen283@wisc.edu

1. Explain your choice of the data structure that you implemented. Did you consider any other data structures besides the one that you implemented? How did you arrive at your final choice of the data structure?
I chose a binary search tree as the data structure, and for each tree nodes I have five fields: two pointers to its children, an integer array recording the position of a word for each of its occurrences, and a char array to store the word. I have thought about using heap or red black tree before. The reason to use binary search tree is that I have not written red black tree and it may be too difficult to compare the priority of strings in a heap.

2. What is the best, average, and worst case complexity of your implementation of the locate command in terms of the number of words in the file that you are querying? (you need to provide all three - best, average, and worst-case analysis). For the complexity, we are only interested in the big-Oh analysis.
For a file with n words, the best, average case complexity is O(log n), and worst case is O(n). For a balance binary tree, the complexity should be O(log n), so the best and average case is O(log n). In the worst case, the tree could looks actually like a linked list because when insertion the words are already sorted, so that the locate complexity can be O(n).

3. What is the average case space complexity of your data structure in terms of the number of words in
the input file? In other words, using the big-Oh notation, what is the expected average size of your
data structure in terms of the number of words.
For n distinct words in the file, we need to allocate n nodes to record these words, so the space complexity is O(n).