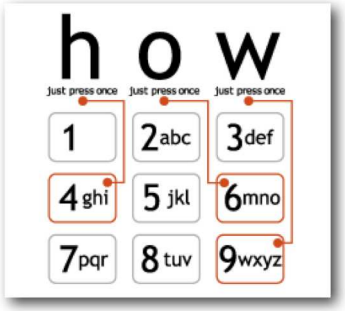
COS226 Week 9 Group Activity

	charAt() calls (typical case)			
implementation	search hit	search miss	insert	space (references)
red-black BST	L + lg 2 N	lg 2 N	lg 2 N	4 N
hashing (linear probing)	L	L	L	4 N to 16 N
R-way trie	L	lg N	L	(R + 1) N
TST	L + lg N	lg N	L + lg N	4 N

The values in this table only apply for a very specific string model and model of computation. They should be used as a rough guide only! Do not memorize or write on a cheat sheet. What matters is which entries are linear and which are sublinear.



www.t9.com

1. Tries

   (a) Design a symbol table for handling T-9 texting.

      i. What are the keys?

      ii. What are the values?

      iii. Which string symbol table implementation is best? If using a trie, what is your alphabet?

      iv. If you get done early, which of the symbol table methods below would be useful? How?

      - get()
      - keysWithPrefix(String s)
      - keysThatMatch(String s) //wildcard matching
      - longestPrefixOf(String s)

   (b) For Wordnet, suppose you'd used a Trie to map from noun to synset IDs. What would be the alphabet for our keys? Do you think performance would have been better than using an LLRB/hash table?

   (c) Suppose you'd used a Trie to map from synset IDs to synset string (for example, 0 → "juvenile juvenile_person"). What would be the alphabet for our keys?

   (d) Other than performance, what is the benefit of a trie over LLRB and hash based symbol tables?

(e) When would you use a hash table over a trie?

2. Sorting

(a) Show the results after processing the two rightmost characters for the LSD algorithm. Show the results after processing the two leftmost characters for MSD. For this input, which of these two algorithms will examine more characters?

```
ZORSE
HORSE
BLERG
BLARG
ZERGS
```

(b) Fill in the tables below, but not during precept.

	total #charAt() calls	
algorithm	worst case	random string case
LSD		
MSD		

		#charAt() calls / compare		overall run time	
algorithm	# string compares	worst case	random string case	worst case	random string case
mergesort			lg N		

(c) What sort of input is best for MSD and mergesort? Worst?

(d) For what sort of input would LSD perform better than MSD?

14. Legume Grime Pop (**14 points).**

Congratulations, you've graduated, and have found employment at Peg Leg Emporium, providing the highest quality peg legs at the lowest legal prices.

Sadly, the peg leg business is really hurting, and your boss wants a truly memorable advertising slogan. To this end, she proposes the following task: Design an algorithm for finding the largest **anagram set** in the English language. An **anagram set** is a set of words such that all words are anagrams of one another. For example, {tars, arts, rats, star} is an anagram set.

Your algorithm should run in NL^2 time, where L is the length of the longest English word, and N is the number of words. You may assume you have a text file containing all English words. You do not need to describe how to read in this text file, nor do you need to count reading this text file in your runtime.

Your program should print all members of the largest anagram set to the screen. If there is more than one, you may print any of them. You may use any algorithm discussed in class.

(a) Describe an algorithm that finds the largest anagram set (hint on the next page). List any data structures that your algorithm needs, and write the order of growth of the run time of every step of your algorithm in terms of N and L.

That hint I promised: Start by finding ALL anagram sets.

(b) Your marketing campaign was a success, and you got a pretty good high five at last week's meeting, but now the boss has an even crazier idea, slyly mentioning that you might just be able to earn Employee of the Month. Now she wants the longest **anagram ladder** in English. An **anagram ladder** is a sequence of words such that the $k+1^{th}$ word is an anagram of the k^{th} word plus any character in the English alphabet. For example {to, lot, lost, toils, tonsil, lotions, colonist, locations, coalitions, dislocation, conditionals, consolidation, consolidations} is an anagram ladder.

Provide an algorithm (and precisely list any data structures you use) that produces the longest anagram ladder in English. Write the runtime of every step of your algorithm, as well as the total runtime**.** If you built some data structure in part a that is useful to you here, you may reuse it. For full credit complete this task in order of growth NL^2 time or better in the worst case.