Computer Science 211 PA 3 Due April 30, May 4
No late days past May 4

Part I

1. Reimplement the CS211CountingDictionary interface using a 26-way Trie. Start with the code in /home/staff/jerager/public/cs211/pa3. I have implemented one of the major methods for you, make sure you understand it before you try to do the others. You will need to add a method to your Word class to compile and run Trie, which contains a test program. You may need to add other methods a well. If you are doing delete (are you shooting for an A?), you should clean up any dangling references behind you. Make this method recursive and it will be easy. I suggest you write a boolean method called anyKids which tells you whether at least one child pointer is non-null.

Submit this to CS211PA3A by April 30

Part II

The examples in the following refer to this Trie:

```
empty
- A Word: a  1
- empty
-- A Word: he  1
--- empty
---- empty
----- A Word: hello  2
------ A Word: hellor  1
-- empty
--- A Word: him  1
- empty
-- empty
--- empty
---- empty
----- A Word: mezza  1
----- A Word: mezzo  1
- empty
-- empty
--- A Word: why  1
```

2. Write a method in your Dictionary class called prefixMatch(String s) that returns a Vector containing all the words that start with the given string. (This should be a very simple extension of Part 1.)

prefixMatch(mez) should return mezza mezzo

Submit this to CS211PA3B by April 30

Part III

3. Write a method spellCheck1(String aword) to spell check a word. Return a Vector<Word> of all the words that share the longest prefix that matches aword in the dictionary.

spellCheck1("hezzo") should return he hello hellor
spellCheck1("hazzo") should return he hello hellor him

Submit this to CS211PA3C by May 4

Part IV (A level assignment)

4.Spell check a word in a different way. Define the distance between two equal-length words as the number of letters in which they differ (comparing character by character). For example,

- `place` and `peace` have distance 1 and
- `place` and `plank` have distance 2.

Given a (possibly misspelled) target string and a maximum distance, write a method spellCheck2(String target) that gathers from the dictionary **the Vector of Words** that have a distance to the target string less than or equal to the given `maxDistance`. The returned vector should contain all words in the lexicon that are the same length as the target string and are within the maximum distance. (Note- you should not do this by checking every word in the dictionary. This should be done with recursion.)

Possible results

Target	**Max distance**	**Suggested corrections**
`ben`	1	`zen`
`nat`	2	`new, not`
`crw`	1	`caw, cow, cry`
`zqwp`	2	`gawp, yawp`

spellCheck2("hezzo",2) should return mezza, mezzo, hello
spellCheck2("hezzo",1) should return mezzo

5. Add at least one other interesting method to the Trie class. Do what you want – add other methods to aid searches or spell checks. Be sure to describe this in the comments of the Trie class.

Submit this to CS211PA3D by May 2