HW 5: Frequency Counting with a Red-Black Tree

On the surface, this will be a repeat of the last assignment. However, it will be backed using a red-black tree rather than a hash table.

Your main program will be called WordFreqs2. As before, the command-line argument is the file to open. It will count the word frequencies as before, and then give a prompt asking the user for words, for which it will give the frequency and other information.

The difference is the addition of a few more commands:

- `< and >` will get the first key and the last key, respectively.
- `<word` and `>`word will get the predecessor and successor of the word in question.
- Extra credit: `&word` will get the rank of a word, while `*rank` will get the word with integer rank given.

To solve this problem, you must will implement a red-black tree. It will need the following functions. All these functions must run in logarithmic time, unless otherwise noted.

- A constructor. No arguments are needed.
- `put(key, value)` which maps key to value in the table.
- `get(key)`, which returns the value to which key maps.
- `contains(key)`, which returns whether or not key is a valid key in the table.
- `delete(key)`, which removes key and its value from the table.
- `size()`, which returns the number of elements the hash table contains in constant time.
- `isEmpty()`, which returns true if the structure is empty, in constant time.
- `getMinKey()`, which returns the first key (according to their order).
- `getMaxKey()`, which returns the last key (according to their order).
- `findPredecessor(key)`, which returns the key before the given one (or null/None if the given key was the first, or if the given key is not in the tree).
- `findSuccessor(key)`, which returns the key after the given one according to their order (or null/None).
- `findRank(key)` which determines the key’s rank (extra credit).
- `select(rank)` which returns the key with integer rank rank (extra credit).

If you use Java, generics are again required. If you are not using Java, you are responsible for implementing similar behavior with respect to the objects.
Words should still be defined as follows: they only consist of A-Z, a-z, 0-9, _, and '. However, '
' is not a legal beginning or end of a word. Any word that begins or ends with a ' has it
repeatedly removed, until it is no longer there.

Thus, a sample run might proceed like this:

```
user@computer ~/files/algo/hw $ java WordFreqs2 alice.txt
This text contains 2629 distinct words.
Please enter a word to get its frequency, or hit enter to leave.
> alice
"alice" appears 386 times.
> <
The alphabetically-first word in the text is "a".
> <alice
The word "alas" comes before "alice".
> >
The alphabetically-last word in the text is "zigzag".
> >alice
The word "alice’s" comes after "alice".
>
Goodbye!
```