CS 61B Spring 2023

Iterators, Polymorphism

Exam-Level 5: February 13, 2023

1 Take Us to Your "Yrnqre"

Kyle, an intrepid space explorer, has just landed on a strange alien planet and has to communicate that he means no harm! Luckily, these aliens use the same alphabet as the English language, but in a different order.

Given the `AlienAlphabet` class below, fill in `AlienComparator` class so that it compares strings lexicographically, based on the order passed into the `AlienAlphabet` constructor. For simplicity, you may assume all words passed into `AlienComparator` have letters present in `order`.

For example, if the alien alphabet has the order `"dba..."`, which means that `d` is the first letter, `b` is the second letter, and so on. `AlienComparator.compare("dab", "bad")` should return a value less than 0, since `dab` comes before `bad`.

If one word is an exact prefix of another, the longer word comes later. For example, `"bad"` comes before `"badly"`.

Hint: You may want to use the `indexOf` method.

```
1  public class AlienAlphabet {
2      private String order;
3
4      public AlienAlphabet(String alphabetOrder) {
5          order = alphabetOrder;
6      }
7
8      public class AlienComparator implements Comparator<____________> {
9          public int compare(String word1, String word2) {
10
11             int minLength = Math.min(_________________________, _________________________);
12
13             for (_____________________________________________________________________) {
14
15                 int char1Rank = ________________________________________________;
16
17                 int char2Rank = ________________________________________________;
18
19                 if (_____________________________________________) {
20                     return -1;
21
22                 } else if (_____________________________________________) {
23                     return 1;
24                 }
```

```
25              }
26
27              return _________________________________ - _________________________________;
28          }
29      }
30  }
```

2 Iterator of Iterators

Implement an `IteratorOfIterators` which will accept as an argument a `List` of `Iterator` objects containing `Integers`. The first call to `next()` should return the first item from the first iterator in the list. The second call to `next()` should return the first item from the second iterator in the list. If the list contained `n` iterators, the `n+1`th time that we call `next()`, we would return the second item of the first iterator in the list.

Note that if an iterator is empty in this process, we continue to the next iterator. Then, once all the iterators are empty, `hasNext` should return **`false`**. For example, if we had 3 `Iterators` A, B, and C such that A contained the values `[1, 3, 4, 5]`, B was empty, and C contained the values `[2]`, calls to `next()` for our `IteratorOfIterators` would return `[1, 2, 3, 4, 5]`.

```
1  import java.util.*;
2  public class IteratorOfIterators  ______________________________ {
3
4
5      public IteratorOfIterators(List<Iterator<Integer>> a) {
6
7
8
9
10
11
12
13
14
15     }
16
17     @Override
18     public boolean hasNext() {
19
20
21
22
23
24
25
26     }
27
28
29
30     @Override
31     public Integer next() {
32
33
```

```
34
35
36
37
38
39          }
40      }
```