HackerRank For Work

INFO6205-05 Spring 2018 Fi… 75 minutes

Question - 1
Linear Probing Hash Table

SCORE: **30 points**

Consider the *delete()* method for a linear probing hash table.
Issues to be solved:
** For simplicity, DO NOT worry about resizing. The initial hash table size is set to 16 and it doesn't need to be changed.

a. You should be able to remove a key;
b. You should remove the value as well to prevent loitering;
c. When you remove a key/value pair, the following key/value pairs should be adjusted as appropriate;
d. Your code should also deal with wrapped key/value pairs. That is, if the index pointer reaches the size boundary (=16), it should go back to 0.

Code quality matter. Code should by DRY (Don't repeat yourself).

Question - 2
Frequency Counter

SCORE: **20 points**

You have 2 classes. **FrequencyCounter.java** and **FrequencyCounterTest.java.** FrequencyCounter.java implements a map as the underlying data structure. Your task is to implement 2 methods ***public int get(Object key)*** and ***public void increment(K s).*** As the name suggests FrequencyCounterTest.java contains the unit tests for FrequencyCounter.

FrequencyCounter.java
- Maintains a frequency of the keys used.
- The key of this class is of parametric type <K>. The type of the value is Integer.
  - You are **NOT** allowed to change anything in this class apart from implementing the 2 functions. They have been marked with TODO.
- This class does contain a main method. The main method is purely there to safe guard against clicking the **Run Main()** button. I see no reason to why you should use the main, the unit tests will test your logic. Although if you feel you need to see some output for the sake of debugging, you may add some print statements here. In the event you do use the main, **PLEASE REMOVE** all the code **YOU** added to the main.
- You **MUST** run the unit tests.

FrequencyCounterTest.java
- Contains the unit tests for the above class.
- **Do Not** change anything in this class.

public void increment(K s)
- This function returns nothing, as you can see from the signature ***void.***
- This function stores the frequency of the key provided to the method. The function will essentially check the map for the value of the key and increment it by one.

public int get(Object key)
- Gets the value associated with key from the underlying map.
- value you return will be the frequency of the key inserted. That is, how many times that key has been used.

```
Note: This is a high level example to help
understand what is required and not an exact
working of the code.

Suppose the user has executed the below steps:
 1.increment(0);
 2.increment(1);
 3.increment(2);
 4.increment(1);

The map formed will resemble the below mapping.
The map in this case is a map<Integer,Integer>.
Key -> Value
0   ->  1
1   ->  2
2   ->  1

So if we invoke:

get(0), you should return 1.
get(1), you should return 2.
get(2), you should return 1.

Lets say the user then decides to invoke the
increment function.

increment(0);
increment(2);

The resulting map will look like the following:
Key -> Value
0   ->  2
1   ->  2
2   ->  2
```