Programming Assignment 9

Due 23 NOV @ 11:59pm

Write a program that implements the provided BasicSymbolTable API interface using a hash table that resolves collisions using separate chaining. Skeleton implementations of the hash table (ChainingSymbolTable), linked list (EntryList), and key (ProductID) are provided **where you need to add your code**.

You must properly implement the API. This includes not allowing null keys, null values values, and no duplicates. You have to keep track of the size of each linked list. The ChainingSymbolTable will also have to maintain the size of the overall hash table. Note: you cannot derive the hash table's size from the linked list, the size() method must remain O(1) operation.

You should shrink/rehash when the load factor falls below the minimum threshold of 2 but should not allow M to go below the initial value of 4. You should expand/rehash when the load factor goes above the maximum threshold of 8. You should use the constants provided.

You will also have to implement the hashCode and equals methods for the provided ProductID object to serve as the key for the symbol table. The implementations should use the approaches outlined in lecture. You do not need to modify the Product class.

Implementation Notes

ChainingSymbolTable:

- Use array of EntryList. Need to use cast as follows:
  `this.items = (EntryList<Key,Value>[]) new EntryList[initialM];`
- Size is synonymous with N, items.length is synonymous with M.

ChainingSymbolTable.put():

- Increment size if necessary. Rehash if necessary.

ChainingSymbolTable.delete():

- Decrement size if necessary. Rehash if necessary.

ChainingSymbolTable.hashFunction(Key key):

- Can be accomplished in several ways. To get similar results to example output, use approach in lecture taking the hashCode from the key, make it positive by &'ing with 0x7fffffff, and then taking modulo the length of the table (i.e. items).

ChainingSymbolTable.keys():

- Use the provided Queue implementation which implements Iterable.

ChainingSymbolTable.rehash():

- Can be accomplished in several ways. Can try creating new ChainingSymbolTable, add all key/value from old table, then set items to new table's items.

EntryList:

- The internal Entry class must be a singly linked list with reference only to front.

EntryList.put(Key key, Value value):

- Must overwrite value if entry for key already exists.
- Push the new, non-duplicate entry onto the front of the list.

EntryList.delete(Key key):

- Keep reference to prior entry so can easily remove. Handle special cases.

Grading Notes

You must:

- Use the template provided for you
- Have a style (indentation, good variable names, etc.)
- Comment your code well (no need to over do it, just do it well)

You may not:

· Make your program part of a package.
· Use *code* from anywhere except your own brain.

Submission Instructions:

· Name a folder with your gmu username
· Put your java files in the folder (but not your .class)
· Zip the folder (not just the files) and name the zip "username-pa9.zip"
· Submit to blackboard

Grading Rubric

No Credit:

· Non-submitted assignments
· Late assignments
· Non-compiling assignments
· Non-independent work

1pt	Submission Format
1pt	Style and Comments
1pt	EntryList put
2pts	EntryList delete
1pt	EntryList keys
1pt	ProductID hashCode, equals
1pt	ChainingSymbolTable put
1pt	ChainingSymbolTable delete
0.5pt	ChainingSymbolTable rehash
0.5pt	ChainingSymbolTable keys

Example Run

```
> java ChainingSymbolTable operations.txt
isEmpty?=true
size=80
size=75
size=75
size=72
size=30
Final mappings=[
GMU,54->GMU,54,student54,30,0.9,
GMU,70->GMU,70,student70,30,0.9,
GMU,62->GMU,62,student62,30,0.9,
GMU,78->GMU,78,student78,30,0.9,
GMU,55->GMU,55,student55,30,0.9,
GMU,71->GMU,71,student71,30,0.9,
GMU,63->GMU,63,student63,30,0.9,
GMU,79->GMU,79,student79,30,0.9,
GMU,56->GMU,56,student56,30,0.9,
GMU,72->GMU,72,student72,30,0.9,
GMU,64->GMU,64,student64,30,0.9,
GMU,80->GMU,80,student80,30,0.9,
GMU,57->GMU,57,student57,30,0.9,
GMU,73->GMU,73,student73,30,0.9,
GMU,65->GMU,65,student65,30,0.9,
GMU,58->GMU,58,student58,30,0.9,
GMU,74->GMU,74,student74,30,0.9,
GMU,66->GMU,66,student66,30,0.9,
GMU,59->GMU,59,student59,30,0.9,
GMU,75->GMU,75,student75,30,0.9,
GMU,51->GMU,51,student51,30,0.9,
GMU,67->GMU,67,student67,30,0.9,
GMU,60->GMU,60,student60,30,0.9,
GMU,76->GMU,76,student76,30,0.9,
GMU,52->GMU,52,student52,30,0.9,
GMU,68->GMU,68,student68,30,0.9,
GMU,61->GMU,61,student61,30,0.9,
GMU,77->GMU,77,student77,30,0.9,
GMU,53->GMU,53,student53,30,0.9,
GMU,69->GMU,69,student69,30,0.9]
Final symbol table=
[0] size=4:(GMU,54), (GMU,70), (GMU,62), (GMU,78)
[1] size=4:(GMU,55), (GMU,71), (GMU,63), (GMU,79)
[2] size=4:(GMU,56), (GMU,72), (GMU,64), (GMU,80)
[3] size=3:(GMU,57), (GMU,73), (GMU,65)
[4] size=3:(GMU,58), (GMU,74), (GMU,66)
[5] size=4:(GMU,59), (GMU,75), (GMU,51), (GMU,67)
[6] size=4:(GMU,60), (GMU,76), (GMU,52), (GMU,68)
[7] size=4:(GMU,61), (GMU,77), (GMU,53), (GMU,69)
Load factor=3.75
```