

Programming assignment 5: Map ADT with hash tables

Bonus 5% for a correct implementation that has no unnecessary repetition of code

30%

Implement the class **Bucket** with a *singly-linked list*.

The class must fully implement the **Map** ADT, including the following operations:

- **insert(key, data)**
 - Adds this value pair to the collection
 - If equal key is already in the collection, **raise ItemExistsException()**
- **update(key, data)**
 - Sets the data value of the value pair with equal **key** to **data**
 - If equal key is not in the collection, **raise NotFoundException()**
- **find(key)**
 - Returns the **data** value of the value pair with equal **key**
 - If equal key is not in the collection, **raise NotFoundException()**
- **contains(key)**
 - Returns **True** if equal **key** is found in the collection, otherwise **False**
- **remove(key)**
 - Removes the value pair with equal **key** from the collection
 - If equal key is not in the collection, **raise NotFoundException()**
- **__setitem__(self, key, data)**
 - Override to allow this syntax:
 - `some_hash_map[key] = data`
 - If equal **key** is already in the collection, update its **data** value
 - Otherwise add the value pair to the collection
- **__getitem__(self, key)**
 - Override to allow this syntax:
 - `my_data = some_bucket[key]`
 - Returns the **data** value of the value pair with equal **key**
 - If equal key is not in the collection, **raise NotFoundException()**
- **__len__(self)**
 - Override to allow this syntax:
 - `length_of_structure = len(some_bucket)`
 - Returns the number of items in the entire data structure

50%

Implement the class **HashMap** with a hash table data structure, using an indexable collection (array or python list that can retrieve a value at a given *index* in $O(1)$ time) of **Bucket**.

The class must fully implement the **Map** ADT, including the following operations:

- **insert(key, data)**
 - Adds this value pair to the collection
 - If equal key is already in the collection, **raise ItemExistsException()**
- **update(key, data)**
 - Sets the data value of the value pair with equal **key** to **data**
 - If equal key is not in the collection, **raise NotFoundException()**
- **find(key)**
 - Returns the **data** value of the value pair with equal **key**
 - If equal key is not in the collection, **raise NotFoundException()**
- **contains(key)**
 - Returns **True** if equal **key** is found in the collection, otherwise **False**
- **remove(key)**
 - Removes the value pair with equal **key** from the collection
 - If equal key is not in the collection, **raise NotFoundException()**
- **__setitem__(self, key, data)**
 - Override to allow this syntax:
 - **some_hash_map[key] = data**
 - If equal **key** is already in the collection, update its **data** value
 - Otherwise add the value pair to the collection
- **__getitem__(self, key)**
 - Override to allow this syntax:
 - **my_data = some_hash_map[key]**
 - Returns the **data** value of the value pair with equal **key**
 - If equal key is not in the collection, **raise NotFoundException()**
- **__len__(self)**
 - Override to allow this syntax:
 - **length_of_structure = len(some_hash_map)**
 - Returns the number of items in the entire data structure

When the number of items in the HashMap has reached 120% of the number of buckets (length of array or list) it must **rebuild()**, doubling the number of buckets.

20%

*This assignment part is not directly connected with the previous parts (**Bucket** and **HashMap**). This class should not be used in any way in those implementations. As a part of testing those parts, this key class could be used as a key type sent into those implementations, but as assignments they are completely separate.*

Implement the class **MyHashableKey**, constructed with an integer value and a string value.

Implement the following operations:

- **`__init__(self, int_value, string_value)`**
 - A constructor that takes an integer value and a string value
- **`__eq__(self, other)`**
 - Compares two instances of MyHashableKey and returns True if their values are equal, otherwise False.
- **`__hash__(self)`**
 - Returns a **positive** integer
 - The integer value must be the same for instances that are equal
 - Otherwise can be any integer
 - ***Don't use the built-in hash functions for integers and strings!***
 - *Full marks given if hash value gives fairly even distribution of values*
 - *Zero marks if all values end up in same bucket*
 - **Bonus 5% for 10% best (most even) distributions**
 - *Note that key values can sometimes be very close to each other, or similar, but in those cases may need particularly good distribution.*