CPS305 Lab 6

Topics:

- Hash Tables

Files:

- Lab6.py
- Incorrectly named files will receive a mark of zero.

Submit Files:

- Submit file through D2L
- Your file should not have any print statements in it.
- You will lose a mark for every print statements.
- There is a getName() function in the Lab5.py file.
- Change the output string to your last and first name.
- Your name should be EXACTLY as written on D2L.
- Incorrectly filed out getName function result in a mark of zero.

Lab Description:

In this lab you will implement our own Hash Tables. The hash functions will be given and will implement different collision handling methods: linear probe, double hashing, and separate chaining.

Requirements:

- First, create a method called getName() that returns your name. If this does not work, you will receive a mark of zero.
- You are given a Python Class template. In this template, there 3 classes: MyHashTable, MyChain, MyDouble. The goal of this lab is to implement hash tables with 3 different collision resolution methods: linear probe, double hashing, and separate chaining.
- A class called MyHashTable is given. Implement a constructor that takes two parameters: one is the size of the table to be constructed; the other a function called hash1. The function has1 will be passed in by the tester. Implement a method put(key, data) where it hashes the key using the hash1 method, stores the data in a single list, and then return True if successful, otherwise return False. Nothing should be done in the event of a hash collision - no data can be overwritten, and no secondary storage method should be introduced.
- A class called MyChainTable that extends MyHashTable is given. Implement a constructor that takes two parameters: one is the size of the table to be constructed; the other a function called hash1. Implement a method put(key, data) where it hashes the key using the hash1 method. The chain should be implement using a linked list. Use the Node class for your implementation. The node will store the key and data.
- A class called MyDoubleHashTable that extends MyHashTable is given. Implement a constructor that takes three parameters: the size of the table to be constructed; a function called hash1; and another function called hash2. These functions will be passed in by the tester. You do not have to implement these functions. Implement a method put(key,data) where it hashes the key using has1 method, stores data in a single list, and in the case where there is a collision, use the hash2 function get the offset step. Your implementation should account for if the table is full or when it cannot find a place to insert. These two cases should result in a return of False and True when the insert is successful.

Restrictions:

- You may not use Python's built-in dict to implement your hash tables.

Testing Your Code:

You can test your code with the tester. The tester is not a be all and end all. You should rely on your own testing before using the tester. Note that the lab should be done on Python 3 and the tester only works on Python 3. The tester is named Lab6Tester.py. Your file must be named correctly and be in the same folder of the tester. Here is an example of executing it on the command line (note that the $ is representative of the beginning of the new line).

```
$ python3 Lab6Tester.py
```