COMP 2002
Assignment 4
Fall 2020
Due: November. 18, 2020 @ 11:59pm

Notes to keep in mind for assignment questions:

- *You may use code from the labs, textbook website, etc. Where possible, start with already working code/classes and modify them accordingly.*
- *Sometimes it's easier to start coding from scratch; a good developer knows when to choose between using already existing code and writing their own code.*
- *For questions that ask for pseudo-code, you may also use Python instead. But you have to consider whether implementing the code is worth the extra time required.*
- *Read the Assignment Submission details thoroughly, and make sure to follow all instructions.*

1. (70 points) Implement an in-place Hash Table ADT with the following methods. You are required to use an array of tuples (key, value) as your structure to maintain your data. This means that your structure should be like H = [(k1, v1), (k2, v2), …]).

> ***a. H.put(k, v):*** Associates a value v with a key k by adding the tuple (k, v) to the array. If the key exists, replaces it with the new value.
> ***b. H.get(k):*** Returns the value v associated with key k in the Hash Table.
> ***c. H.remove(k):*** Removes the tuple (k, v) from H. The method should return (k, v).

Implement the three probing (e.g., linear, quadratic, and double hashing) techniques explained in the lectures. **Tip:** When creating an instance of your class, pass an argument to the constructor to define which probe technique will be used to deal with collisions. You should also pass as an argument the value of ***N*.** This value should be used to create the size of your array. Show some simple codes validating that your methods are working correctly.

2. (30 points) Create an experiment to show a collision handling comparison between the three probing methods. Add some random pairs (k, v) to your hash map and **count the number of collisions** for each key. Show an outcome similar to the following:

> **Key value, #Linear probing, #Quadratic probing, #Double hash**
> 1, 6, 4, 2
> …
>
> This line's meaning is that for key value 1, there were 6 collisions with linear probing, 4 collisions for quadratic probing, and 2 for double hashing.

Finally, compare the three probing strategies in terms of collisions. Which function produces more collisions? Why do you think this is happening? Which function produces less, and why? You can add your answers to the Python code as a comment.

Submission Details

For Question #	Submit the following (all in one ZIP file):
1	Python '.py' file containing your code; please document your code with your ideas
2	Python '.py' file containing your code; please document your code with your ideas