## Assignment 3

**Question (10points).** Implementation of hash tables in Python.

1. Implement a hash table using your own hash function. Use Linear Probing to resolve collisions. You are required to implement 4 functions:

- __getitem__(self, key): returns the value corresponding to key in the hash table. Raise a KeyError if the key does not exist. To use this function: call by table[key].
- __setitem__(self, key, value): set a key-value pair. Raise an exception if the hash table is full and the key does not exist in the table yet. To use this function: call by table[key]=value
- __contains__(self, key): returns True if the key is in the table and False otherwise
- hash(self, key): calculates the hash value for the given key, use your own hash function

In the main function, please write your own test functions, use 2-3 test cases to make sure each function works properly. In your report, please explain why you choose this hash function.

2. Download the dictionary files English_small.txt and English_large.txt. For each file, change the hash table size: 200000, 300000, 400000, measure how long it takes to read the file and store into your hash table. In your report, please make a table to summarize the results. Explain why the table size influences the runtime.

3. Implement quadratic probing, double hashing, and linear probing. Compare them with two criteria: 1) number of collisions, 2) average probe length (probe length: how many tries to find an empty slot; average probe length: total probe length divided by total number of items in the hash table. Summarize the comparison into a table or a plot.