Frequencies and Hash-Table

We are processing for a logistic company handling a huge number of parcels every day. The handling a huge number of deliveries required a planning as well. The company has divided the city into different block of locations such as Gulshan Block 4 is considered one location - each location is identified by a number.

At the end of each day, we have to find the number of parcels remaining for each location and a file is maintained for the entrance and exist locations. This information is critical to strategically plan the next day. Whenever a new parcel passes through entrance, it's location ID is stored in a file and a similar file is also maintained at the exit location for any parcel going outside the warehouse. Presently, it is taking too much time therefore it is decided to automate this process by writing a small utility.

We are provided two text files in which each line contains a number representing the Location ID for the parcel. As discussed above, the entrance file contains the Location IDs for parcels received whereas the exit file contains the Location IDs for parcel going for delivery. Our goal is to generate the list of Location IDs with remaining number of parcels.

Implementation

HashTable:

You have to implement the Hash-Table with linear probing. This file will store each location as a key and its frequency as value. The signature and details of required functions are given in the provided code template. You may add other functions if required.

FrequencyTable:

This call is a kind of wrapper for the HashTable class in order to implement the logic of frequency. The signature and details of required functions are given in the provided code template. You may add other functions if required.

FrequencyUtility:

This class is responsible to read the data from the file and store in to FrequencyTable in order to generate frequency for each location.

Code:	`See the provided code template`

Example:

Consider the following files:

For entrance:	**For exit:**
121345	982756
675839	12364
563719	14653
982756	12364
12364	121345
14653	
12364	
121345	
121345	
563719	

The testing code must print the following output (order can vary):
[(121345,2), (675839,1), (563719,2)]