Show your implementations of each of the following:

insert(self,num) - 15pts

The code starts with 3 different hash functions and an object class called OpenHash. Insert is a method in this class. First, we get the position or the index of the number by apply the hash function to the number, then it will check if the number is already in that row of the hash table and if not, it will insert the number to the hash table.

member(self,num) - 15pts

The member method is also doing similar thing, it will find the row list index of the number, check if the number is already existing in the hash table and return True if yes and return No otherwise.

delete(self,num) - 15pts

The delete method also apply the hash function to the number to find the position of the number in the hash table and then check if the number is already in the row. If yes, it will remove the number from the row list.

Show that all pytests pass - 10pts

Explain/Show how you computed the approximated collision rate - 25pts

To computed the collision average rate, we need to find the count for number of times collision occur first. I added a get\_collision method which will look through the rows in the table then count the number of elements of the rows that have more than 1 element. That is where collision occur because we hash a value to an index that already has at least one value in it, which means that row will have more than 1 element.

Then, I use the random\_sequence function to initialize the table with random number between 0 and 5 times of the size. I also have the average collision function to accumulate the collision count for 10 times of hashing and calculate the average. And I print the output in the main

Analysis of which hash function worked best - 20pts

Based on the output, we can see that as the size increase the first hashing function produce the least collision so it will be the best hashing function compared to the other two. Hash3 function hit second place and the worst is hash2 function.