# Assignment 3

Intro:-

The goal of the assignment is to create a hash table to provide fast and repeated access to data. The Hash table is used in the context of a voting station. A 13 digit ID number will be entered and if it matches the ID number of a person in the data, that person’s name will be printed. If it is not found , the user will be told so.

Method:-

The classes used are:

1. person class
2. HashTable class
3. VoterVerifier class
4. RandomNumber class

1. person class

   The person class enables us to create a person object. The person object will have 2 instance variables. One will be the name of the person and the other will be its ID number. The person class contains a toString and equals method. It also has some accessor methods.

2. HashTable class

   The HashTable class contains methods(hash function) that will create a Hash table and add the person objects inside it. A find and insert method was also implemented. The HashTable has a trials() method which finds every element of the Hash table and prints out the number of comparisons made before finding them. The average is also calculated after finding each element.

3. VoterVerifier class

   The VoterVerifier class is the driver class that the user will use. The user will be required to enter the 13 digit ID number that he wishes to verify.

4. RandomNumber class

   The RandomNumber class is used to generate random 13 digit numbers which were used as the ID numbers in the dataset.

More in depth descriptions of the classes are in the javadocs.

## The Hash table:

An array of linked lists was created and the objects were then added. The size of the dataset used was 150. To get a more evenly distributed Hash table the closest prime number after multiplying the size of the dataset by 1.3 was used as the size of the Hash table. The value obtained was 211. Chaining was used as collision resolution mechanism because it is very effective and easy to implement.

When the hash function was used on the ID number, the value obtained was used as the index for the hash function. If the index is empty, a linked list is created and the person added to the list. If the index is not empty, this means that there is already a linked list present. The person is then added to the linked list.

The hash function used was ID % size, where ID is the 13 digit ID number and size is the size of the Hash table, in this case 211. This is why I am using a prime number as the size of the Hash table, as it will have a bigger chance to give different values.

## The interface:

A text based interface was used for this assignment. The user will enter the 13 digit ID number of the person he wishes to check. If the person is on the list, his/her name will be displayed. If the person is not on the list, the user will be told so. The user will keep looking up IDs until he presses 0. When he presses 0, the program is exited.

## Results:

The first 10 lines when running the trials() method was:

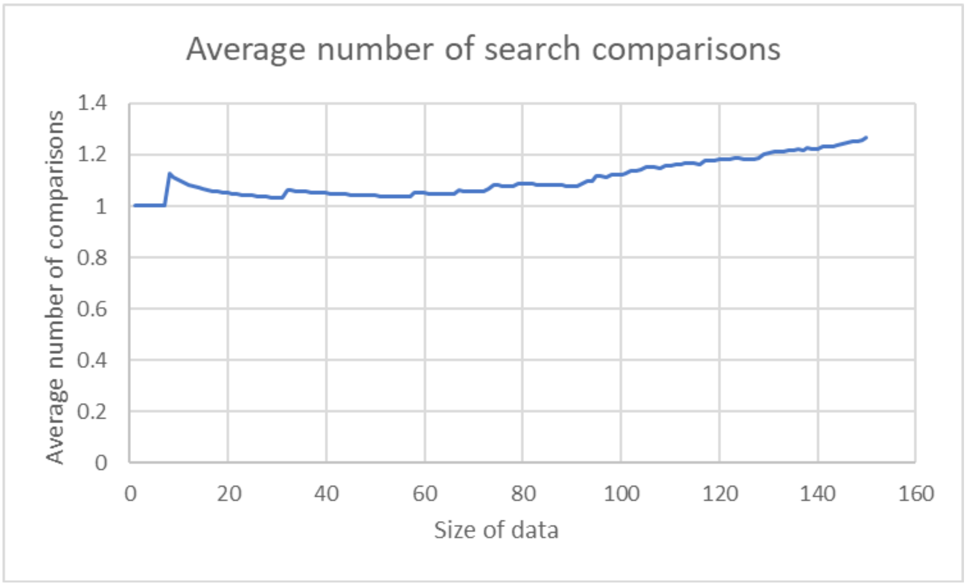
```
Jim Morrison is on the list

Number of comparisons made before finding element was: 1

The current average comparisons is: 1.0
```

```
Gordon Gano is on the list

Number of comparisons made before finding element was: 1

The current average comparisons is: 1.0


Lou Reed is on the list

Number of comparisons made before finding element was: 1

The current average comparisons is: 1.0
```

The last 10 lines obtained  when running the trials() method was:

```
1.234

1.232

1.231

1.236

1.241

1.247

1.252

1.250

1.255

1.267
```

When the number of comparisons by the size of data was plotted, the following graph was obtained:



Here we can see that the Hash table works as expected. A Hash table would ideally offer a runtime complexity of O(1) and in this case, it is very close to this value. The total average value of comparisons made for the total dataset was 1.267. The value is not exactly 1 because there is some collision occurring. When the collision occurs, the data is then added to the linked list. This means that when trying to find the item further in the linked list, the linked list has to be traversed. This increases the number of comparisons made.

The minimum amount of comparisons when searching for a particular element was 1.

The maximum amount of comparisons when searching for a particular element was 3.

The chaining mechanism although not ideal has worked quite well with the average comparisons being close to the best case. This means that when searching for a particular element in the list, the runtime will be nearly O(1).

By contrast a traditional array would offer an average runtime complexity of O(n/2) or in this case, an average of 75 comparisons. This is extremely large compared to the current 1.267.

Creativity:

To generate 13 digit numbers, a program called RandomNumber was created. This is much easier than writing down 150 separate ID numbers.

The trials() method in the HashTable class prints out the number of comparisons when searching for each element in the Hash table.

It also prints out the average comparisons after each search.

It also shows the maximum and minimum search comparisons when searching for a particular a element.

At the end the average after each search is printed out. This was used to plot the graph.

Git log:

```
rahul@rahul-VirtualBox:~/files/Assignment3$ git log

commit 86ffcc4941646b834b9f73f772e7818fa5f60bf0 (HEAD -> master)

Author: Rahul Deeljore <Rahul_2496@hotmail.com>

Date:   Tue Apr 17 12:54:54 2018 +0200


    V1


commit a8cd96f1bc3ba68791ccba408da9a0651a0608e6

Author: Rahul Deeljore <Rahul_2496@hotmail.com>

Date:   Tue Apr 17 12:37:56 2018 +0200


    Final Version


commit 7e44cdeaf77a20de0a94ed0c733c38ea4788ebe9

Author: Rahul Deeljore <Rahul_2496@hotmail.com>

Date:   Tue Apr 17 12:34:16 2018 +0200


    Working version


commit 4cc1b6518c25376d2768bc03fc7db1457222e7bb

Author: Rahul Deeljore <Rahul_2496@hotmail.com>

Date:   Tue Apr 17 12:28:42 2018 +0200


    Nearly Operational version
```

```
commit a6b30755bda1323083089dcb433c61ed6227790d
Author: Rahul Deeljore <Rahul_2496@hotmail.com>
Date:   Tue Apr 17 12:26:58 2018 +0200

    Second version

commit 5a57736c6ce86e9940c39ee454e07d687757682a
Author: Rahul Deeljore <Rahul_2496@hotmail.com>
Date:   Tue Apr 17 12:25:44 2018 +0200

    first version
```