Group 3

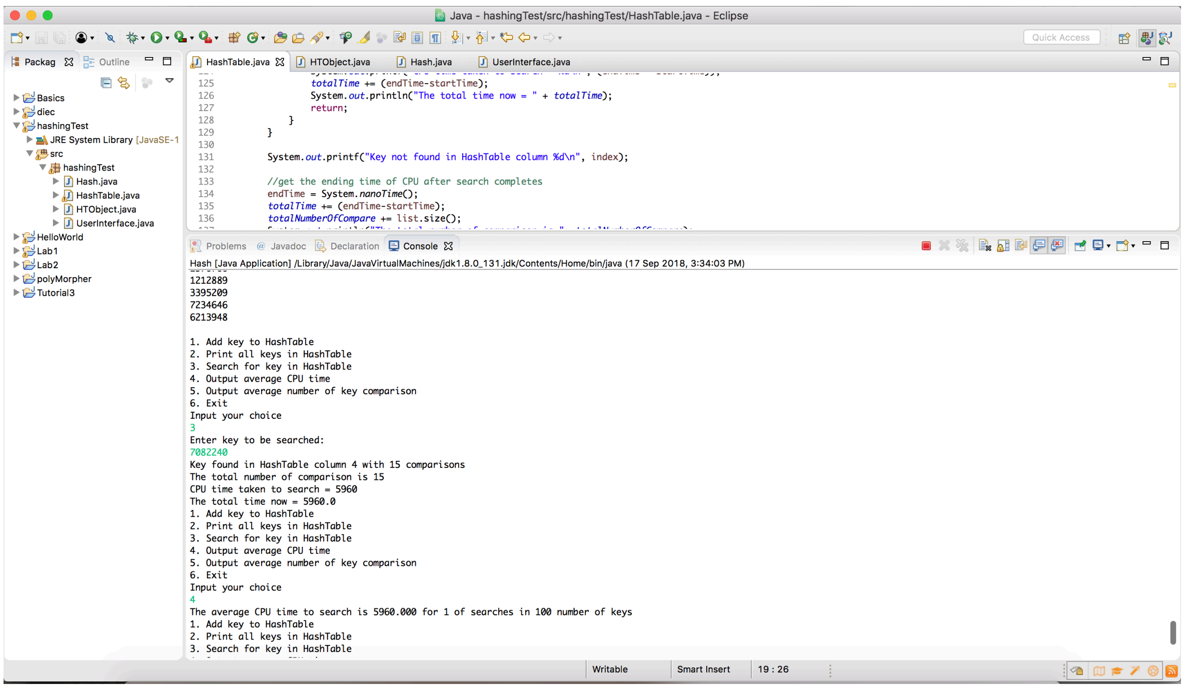
Q2 : Using closed address hashing, compare between hash tables of prime and non- prime sizes.

For closed address hashing, there is a probability of colliding when hashing values that are multiple of the mod table size then chances of colliding is very high. For example, in the case of NRIC it is common that the NRIC is an even number (technically 50% chance). So if the the mod table size is a a even number (non-prime) it will have high chance of colliding. Hence, the number of nodes at a certain number is more saturated than others. Leading to inefficient searching time on average.

Our hashing algorithm is closed address hashing. Meaning it maintains the original based address. Our question require us to compare between prime and non-prime table size hence our mod function will change.

Demo:

<u>**Successful**</u>

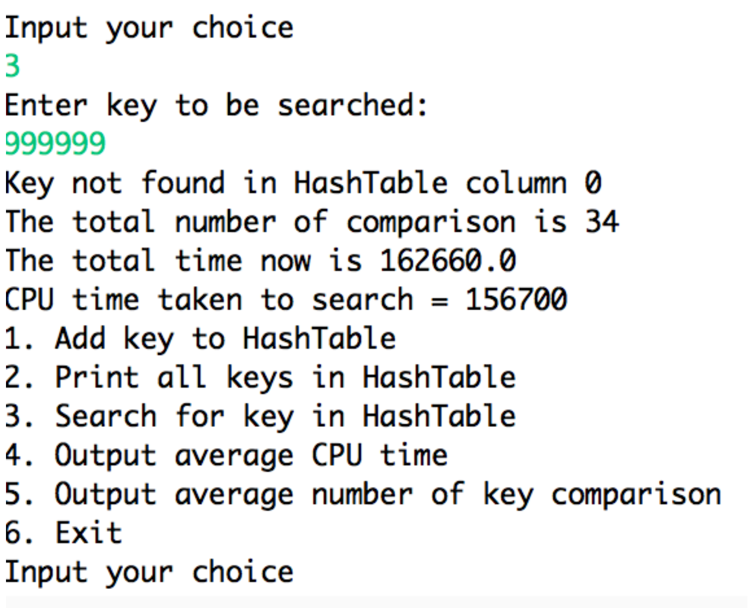


```
                totalTime += (endTime-startTime);
                System.out.println("The total time now = " + totalTime);
                return;
            }
        }

        System.out.printf("Key not found in HashTable column %d\n", index);

        //get the ending time of CPU after search completes
        endTime = System.nanoTime();
        totalTime += (endTime-startTime);
        totalNumberOfCompare += list.size();
```

Hash [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home/bin/java (17 Sep 2018, 3:34:03 PM)

```
1212889
3395209
7234646
6213948

1. Add key to HashTable
2. Print all keys in HashTable
3. Search for key in HashTable
4. Output average CPU time
5. Output average number of key comparison
6. Exit
Input your choice
3
Enter key to be searched:
7082240
Key found in HashTable column 4 with 15 comparisons
The total number of comparison is 15
CPU time taken to search = 5960
The total time now = 5960.0
1. Add key to HashTable
2. Print all keys in HashTable
3. Search for key in HashTable
4. Output average CPU time
5. Output average number of key comparison
6. Exit
Input your choice
4
The average CPU time to search is 5960.000 for 1 of searches in 100 number of keys
1. Add key to HashTable
2. Print all keys in HashTable
3. Search for key in HashTable
```

Unsuccessful



```
Input your choice
3
Enter key to be searched:
999999
Key not found in HashTable column 0
The total number of comparison is 34
The total time now is 162660.0
CPU time taken to search = 156700
1. Add key to HashTable
2. Print all keys in HashTable
3. Search for key in HashTable
4. Output average CPU time
5. Output average number of key comparison
6. Exit
Input your choice
```

CPU TIME:

Searched:
6044629
0507038
8591838
4255154

Searched for unssucessful: 999999, 999998, 999997

Change ARR_Size to change H
LF = 0.25,
H = 400
average CPU time = 3912.750 for 4 searches
The average number of key comparison for searching in hash table is 1.000 for 100 number of key

prime number H = 397
average CPU time = 3821.500 for 4 searches

The average number of key comparison for searching in hash table is 1.000 for 100 number of key

<u>**LF = 0.5**</u>

H = 200
Average CPU time = 6135.20ns for 4 searches
The average number of key comparison for searching in hash table is 1.000 for 100 number of key

Prime number H = 199
The average CPU time to search is 3604.75ns for 4 of searches
The average number of key comparison for searching in hash table is 1.000 for 100 number of key

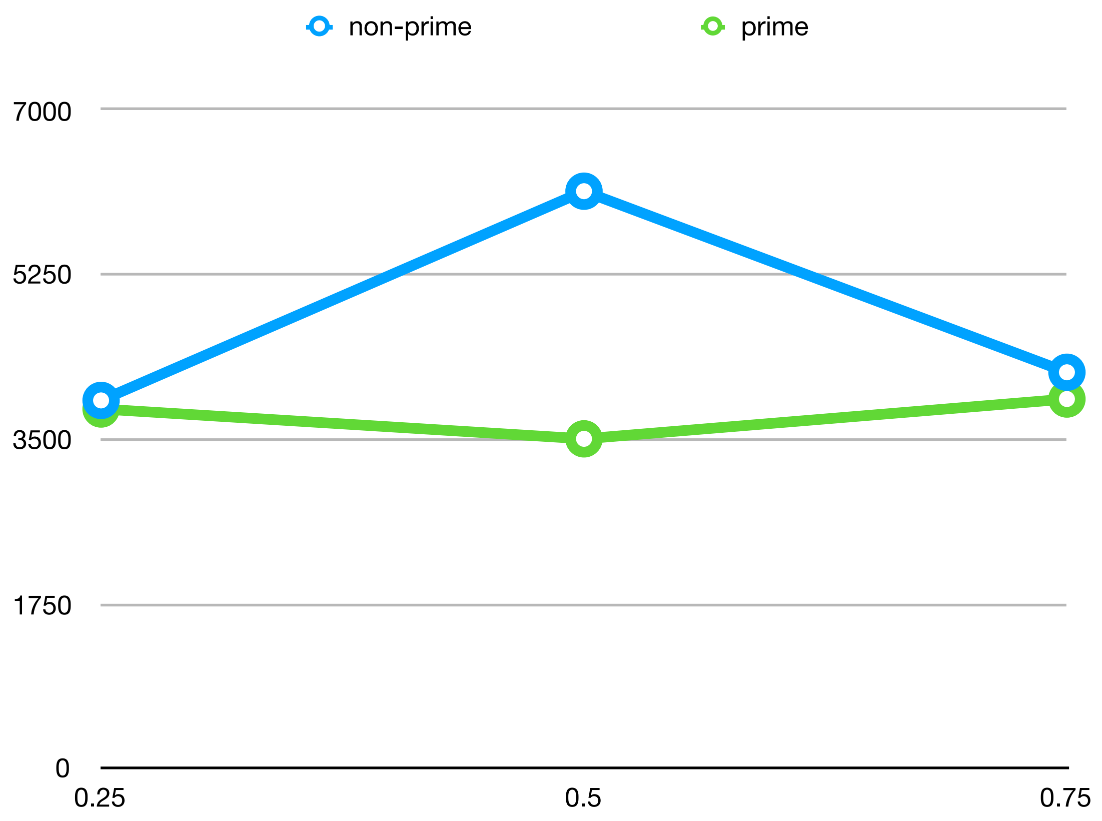
<u>**LF = 0.75,**</u>

H = 133
The average CPU time to search is 4211.00ns for 4 of searches in 100 number of keys
The average number of key comparison for searching in hash table is 1.000 for 100 number of key

Prime number H = 131
The average CPU time to search is 3927.75ns for 4 of searches in 100 number of keys
The average number of key comparison for searching in hash table is 1.000 for 100 number of key

Unsuccessful: CPU TIME = 138244.00
The key number of key comparison is on average 1 due to the low load factor, most of the numbers are first node.

The data is not consistent as java does not have accurate CPU time, however it uses a wall clock which can be inaccurate hence there is anomaly in the graph. However in theory, as load factor increases, average CPU time should increase as seen from H = 400 and H = 200.



Time for prime hashing is faster as the data is better spread out, chances of the data being the first node is higher hence lesser comparison needed therefore faster.

Also in theory the unsuccessful time has to be the same as it has to search throughout all the nodes in the table.