GTU Department of Computer Engineering

CSE 222/505 - Spring 2021

Homework 5 Report

FATİH DOĞAÇ

1901042654

4. PROBLEM SOLUTION APPROACH

1. Only problem that I have seen was to test great values of key-value pairs. Because I couldn’t track them with my eyes. And I couldn’t debug as well. But I was able to figure out a way which is an If else block. That solved my problem by s electing the inputs and outputs that crashes the program and showing them all to me.
2. Implementing a MapIterator that iterates the Map elements with the following features:

   - next(): The function returns the next key in the Map. It returns the first key when there is no not-iterated key in the Map.

   - prev(): The iterator points to the previous key in the Map. It returns the last key when the iterator is at the first key.

   - hasNext(): The method returns True if there are still not-iterated key/s in the Map, otherwise returns False.

   - MapIterator (K key): The iterator should start from the given key and still iterate though all the keys in the Map. The iterator starts from any key in the Map when the starting key is not in the Map or not specified (zero parameter constructor).

3. Implementing 3 HashMaps with the following features:

   - Use the chaining technique for hashing by using linked lists(available in the text book) to chain items on the same table slot.

   - Use the chaining technique for hashing by using TreeSet (instead of linked list) to chain items on the same table slot.

   - Use the Coalesced hashing technique. This technique uses the concept of Open Addressing to find first empty place for colliding element by using the quadratic probing and the concept of Separate Chaining to link the colliding elements to each other through pointers (indices in the table). The deletion of a key is performed by linking its next entry to the entry that points the deleted key by replacing deleted entry by the next entry.

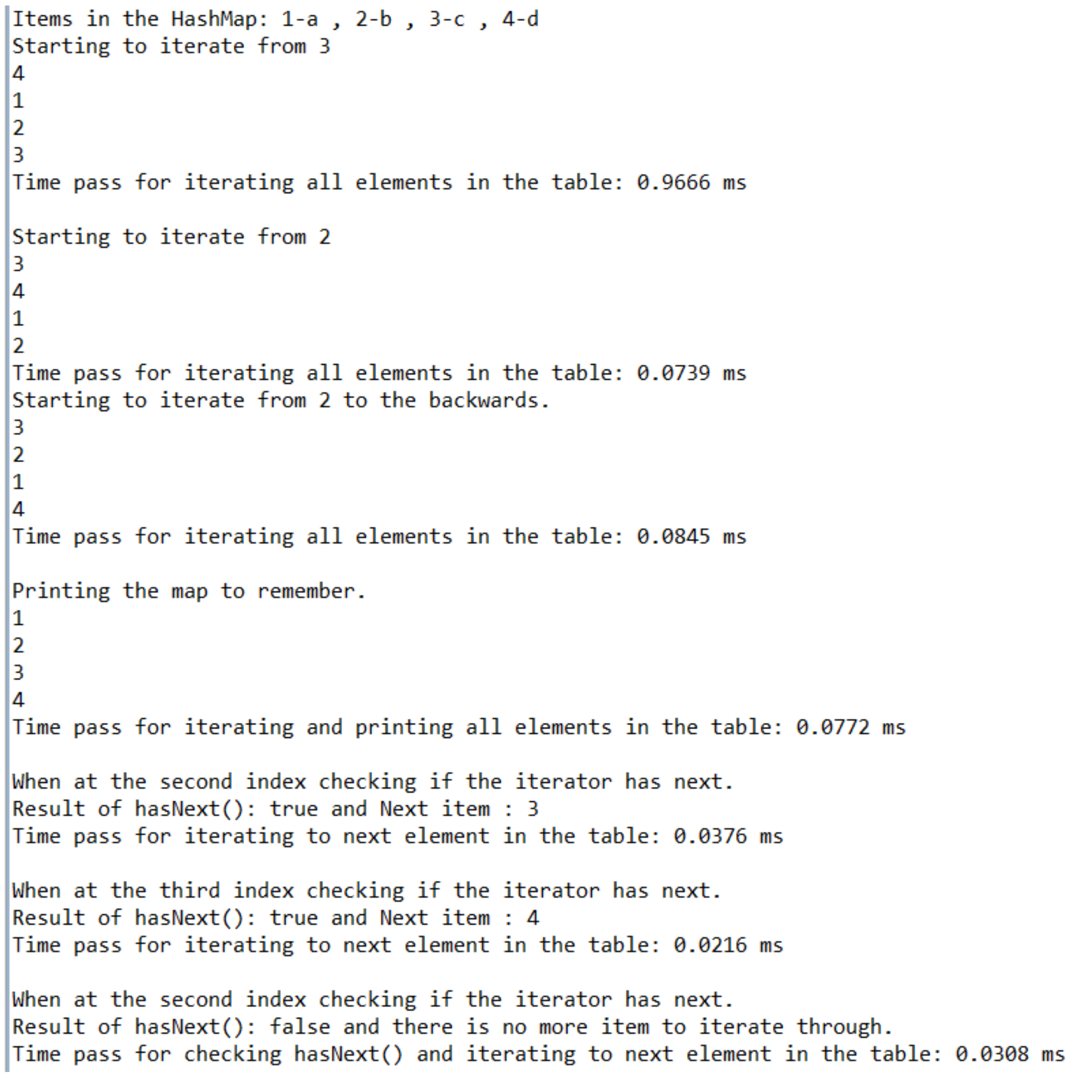
For all of them , first I implemented myself a HashMap with chain technique. After understanding every direction of HashMaps , I thought what should iterator of a map should look like. After lots of researches, I had an idea for homework and I made it real.

Then , I learnt the other techniques of implementing HashMaps and researched every one of them. After lots of debugs my assignment was done.

Test Cases:

There is a driver class in the .tar.gz file to test all the features in the project.

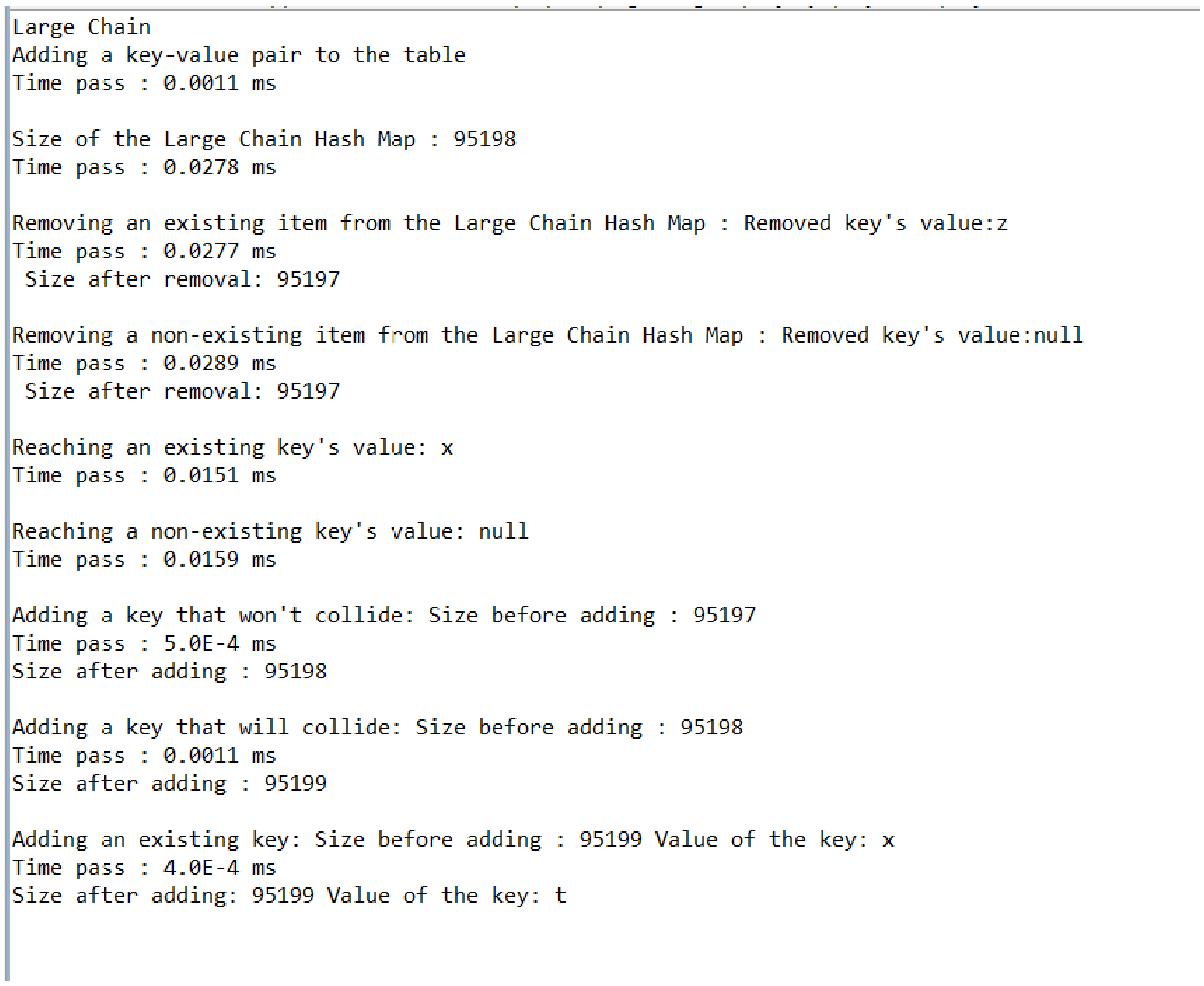
Running and Results:



```
Items in the HashMap: 1-a , 2-b , 3-c , 4-d
Starting to iterate from 3
4
1
2
3
Time pass for iterating all elements in the table: 0.9666 ms

Starting to iterate from 2
3
4
1
2
Time pass for iterating all elements in the table: 0.0739 ms
Starting to iterate from 2 to the backwards.
3
2
1
4
Time pass for iterating all elements in the table: 0.0845 ms

Printing the map to remember.
1
2
3
4
Time pass for iterating and printing all elements in the table: 0.0772 ms

When at the second index checking if the iterator has next.
Result of hasNext(): true and Next item : 3
Time pass for iterating to next element in the table: 0.0376 ms

When at the third index checking if the iterator has next.
Result of hasNext(): true and Next item : 4
Time pass for iterating to next element in the table: 0.0216 ms

When at the second index checking if the iterator has next.
Result of hasNext(): false and there is no more item to iterate through.
Time pass for checking hasNext() and iterating to next element in the table: 0.0308 ms
```

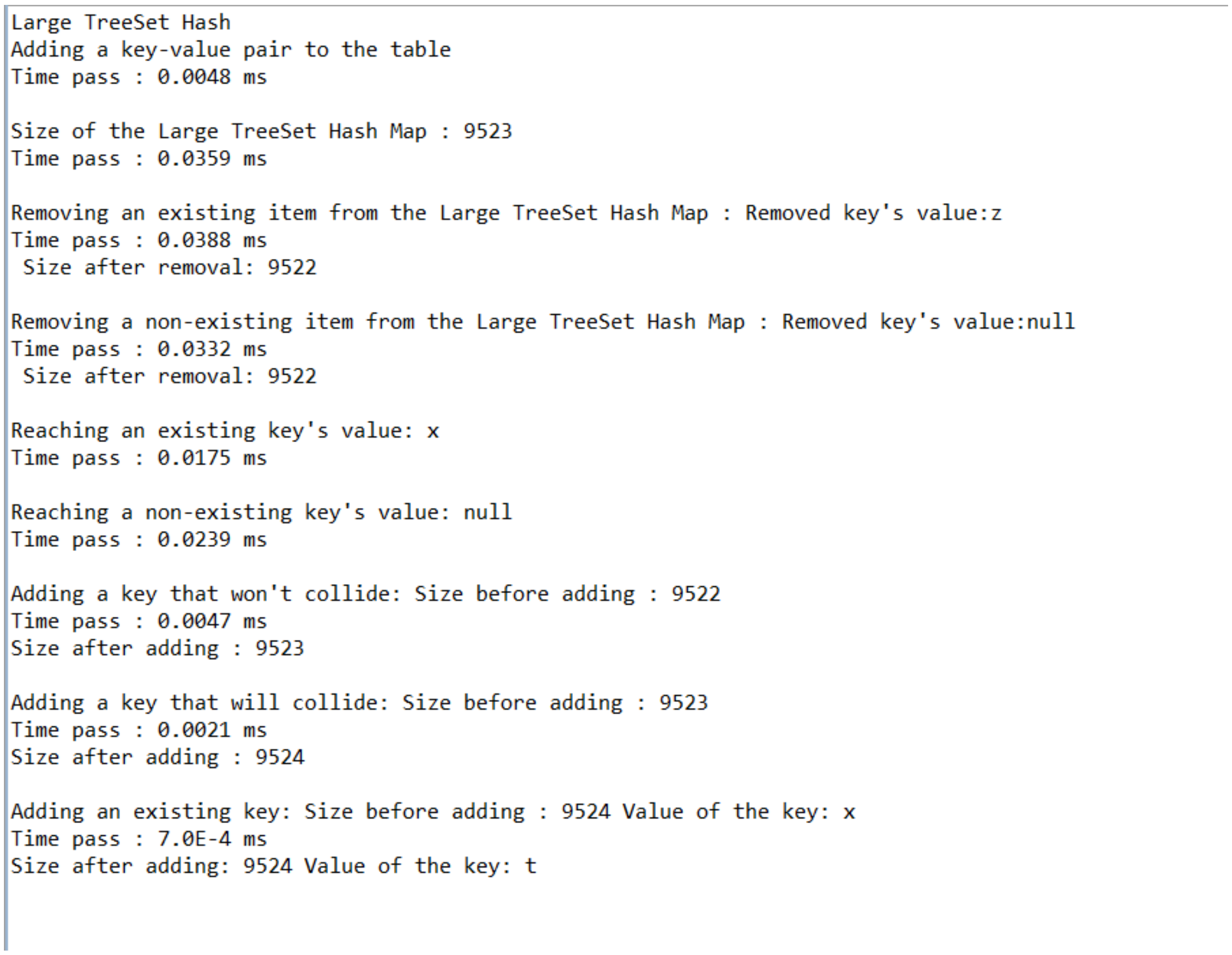
```
---------- PART2 ----------



Small Chain
Adding a key-value pair to the table
Time pass for adding a key-value pair in the table: 0.0084 ms

Size of the Small Chain Hash Map : 9
Time pass for printing the size of the table: 0.0222 ms

Removing an existing item from the Small Chain Hash Map : Removed key's value:q
Time pass for removing an existing item from the table: 0.0459 ms
 Size after removal: 8

Removing a non-existing item from the Small Chain Hash Map : Removed key's value:null
Time pass for removing a non-existing item from the table: 0.0364 ms
 Size after removal: 8

Reaching an existing key's value: x
Time pass for getting an existing key's value from the table: 0.0242 ms

Reaching a non-existing key's value: null
Time pass for getting a non-existing key's value from the table: 0.0193 ms

Adding a key that won't collide: Size before adding : 8
Time pass : 0.0051 ms
Size after adding : 9

Adding a key that will collide: Size before adding : 9
Time pass : 0.0044 ms
Size after adding : 10

Adding an existing key: Size before adding : 10 Value of the key: x
Time pass : 0.0084 ms
Size after adding: 10 Value of the key: t
```

```
Medium Chain
Adding a key-value pair to the table
Time pass : 0.004 ms

Size of the Medium Chain Hash Map : 93
Time pass : 0.0232 ms

Removing an existing item from the Medium Chain Hash Map : Removed key's value:z
Time pass : 0.0258 ms
 Size after removal: 92

Removing a non-existing item from the Medium Chain Hash Map : Removed key's value:null
Time pass : 0.0236 ms
 Size after removal: 92

Reaching an existing key's value: x
Time pass : 0.0173 ms

Reaching a non-existing key's value: null
Time pass : 0.0163 ms

Adding a key that won't collide: Size before adding : 92
Time pass : 0.0103 ms
Size after adding : 93

Adding a key that will collide: Size before adding : 93
Time pass : 0.0037 ms
Size after adding : 94

Adding an existing key: Size before adding : 94 Value of the key: x
Time pass : 0.0024 ms
Size after adding: 94 Value of the key: t
```

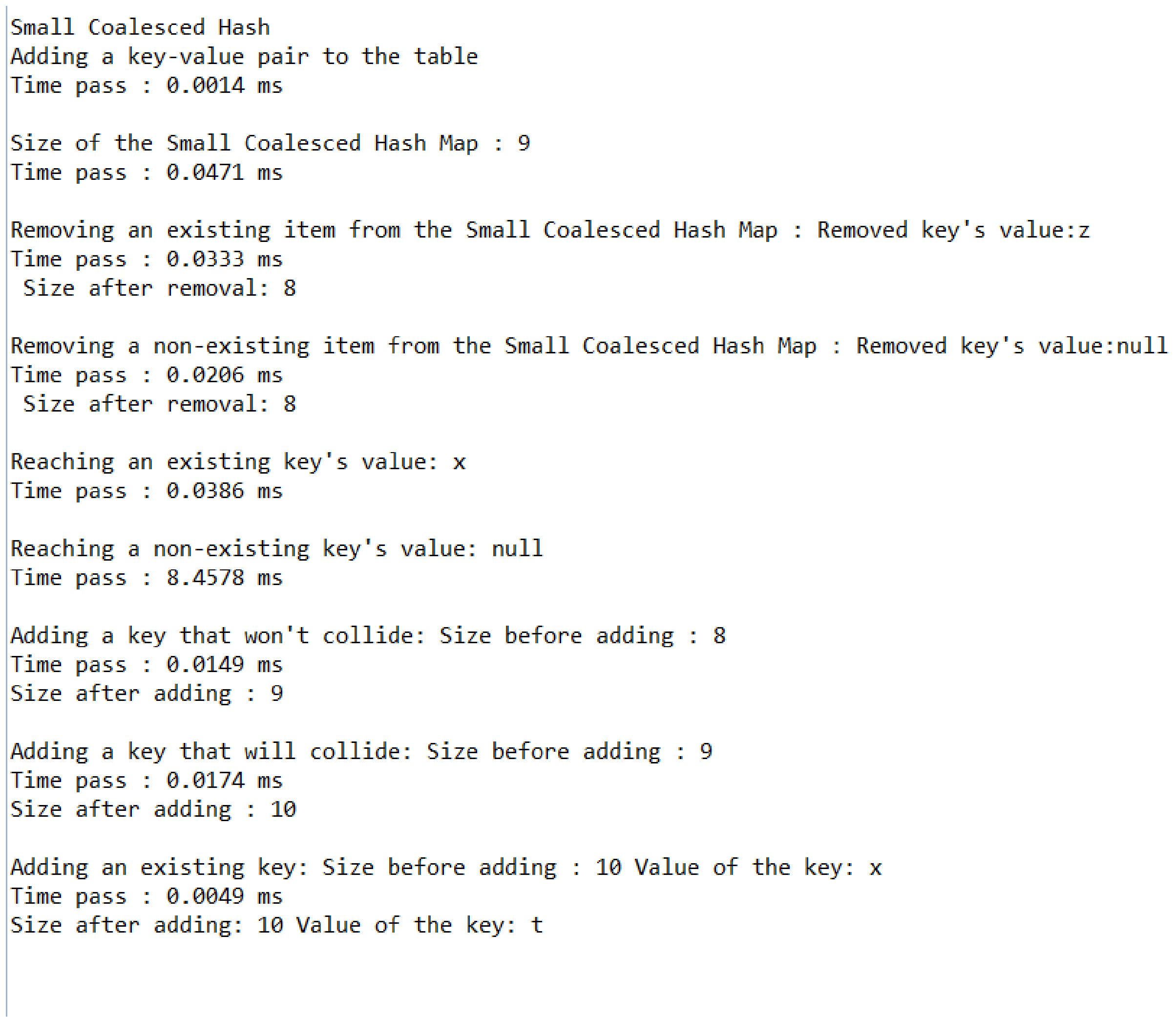

```
Large Chain
Adding a key-value pair to the table
Time pass : 0.0011 ms

Size of the Large Chain Hash Map : 95198
Time pass : 0.0278 ms

Removing an existing item from the Large Chain Hash Map : Removed key's value:z
Time pass : 0.0277 ms
 Size after removal: 95197

Removing a non-existing item from the Large Chain Hash Map : Removed key's value:null
Time pass : 0.0289 ms
 Size after removal: 95197

Reaching an existing key's value: x
Time pass : 0.0151 ms

Reaching a non-existing key's value: null
Time pass : 0.0159 ms

Adding a key that won't collide: Size before adding : 95197
Time pass : 5.0E-4 ms
Size after adding : 95198

Adding a key that will collide: Size before adding : 95198
Time pass : 0.0011 ms
Size after adding : 95199

Adding an existing key: Size before adding : 95199 Value of the key: x
Time pass : 4.0E-4 ms
Size after adding: 95199 Value of the key: t
```

```
Small TreeSet Hash
Adding a key-value pair to the table
Time pass : 0.0067 ms

Size of the Small TreeSet Hash Map : 9
Time pass : 0.0386 ms

Removing an existing item from the Small TreeSet Hash Map : Removed key's value:z
Time pass : 0.0587 ms
 Size after removal: 8

Removing a non-existing item from the Small TreeSet Hash Map : Removed key's value:null
Time pass : 0.0362 ms
 Size after removal: 8

Reaching an existing key's value: x
Time pass : 0.0282 ms

Reaching a non-existing key's value: null
Time pass : 0.034 ms

Adding a key that won't collide: Size before adding : 8
Time pass : 0.0089 ms
Size after adding : 9

Adding a key that will collide: Size before adding : 9
Time pass : 0.0068 ms
Size after adding : 10

Adding an existing key: Size before adding : 10 Value of the key: x
Time pass : 0.0085 ms
Size after adding: 10 Value of the key: t
```

```
Medium TreeSet Hash
Adding a key-value pair to the table
Time pass : 0.0077 ms

Size of the Medium TreeSet Hash Map : 94
Time pass : 0.0688 ms

Removing an existing item from the Medium TreeSet Hash Map : Removed key's value:z
Time pass : 0.0499 ms
 Size after removal: 93

Removing a non-existing item from the Medium TreeSet Hash Map : Removed key's value:null
Time pass : 0.0332 ms
 Size after removal: 93

Reaching an existing key's value: x
Time pass : 0.0204 ms

Reaching a non-existing key's value: null
Adding a key that won't collide: Size before adding : 93
Time pass : 0.0216 ms

Time pass : 0.0075 ms
Size after adding : 94

Adding a key that will collide: Size before adding : 94
Time pass : 0.0031 ms
Size after adding : 94

Adding an existing key: Size before adding : 94 Value of the key: x
Time pass : 0.0029 ms
Size after adding: 94 Value of the key: t
```

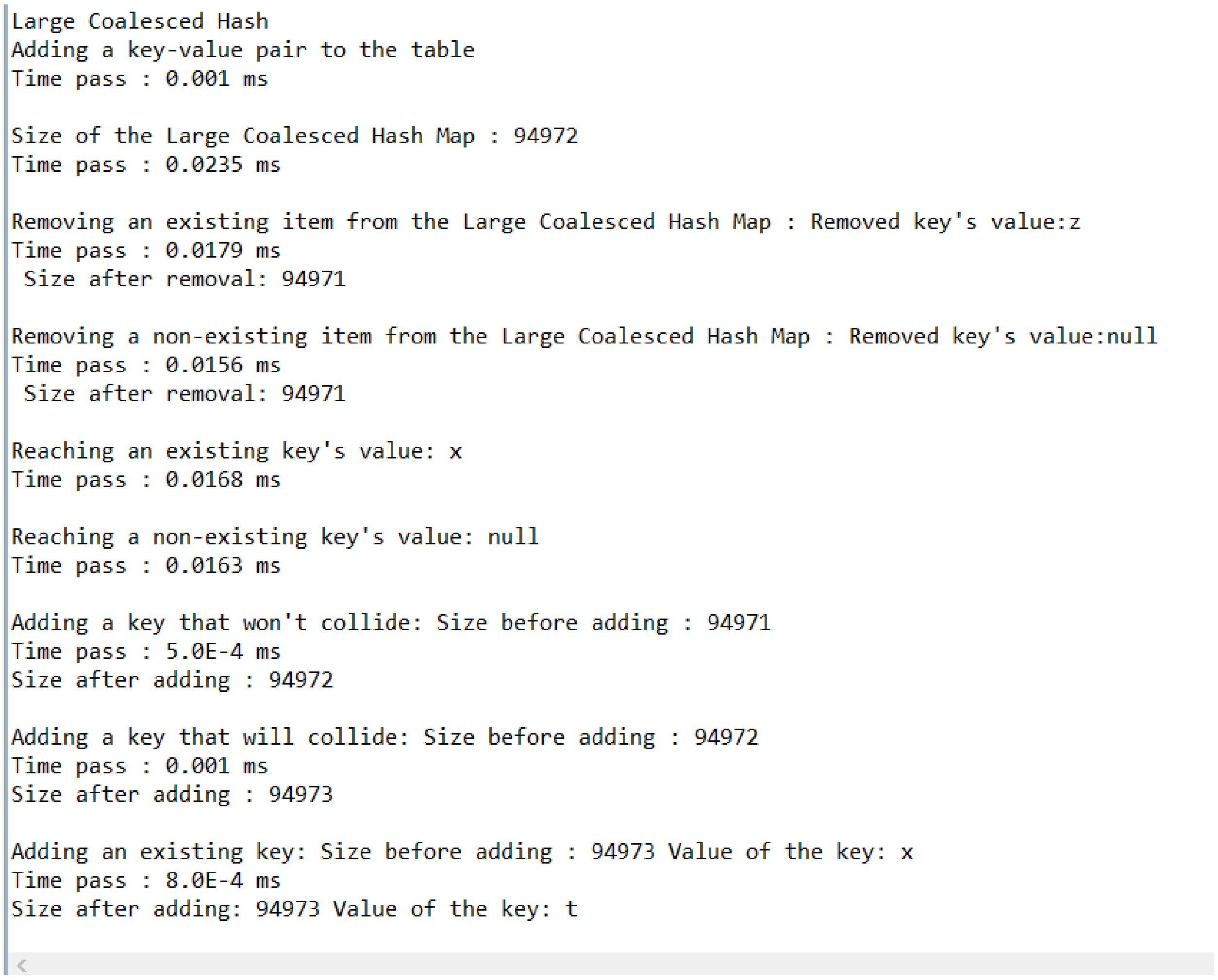


```
Large TreeSet Hash
Adding a key-value pair to the table
Time pass : 0.0048 ms

Size of the Large TreeSet Hash Map : 9523
Time pass : 0.0359 ms

Removing an existing item from the Large TreeSet Hash Map : Removed key's value:z
Time pass : 0.0388 ms
 Size after removal: 9522

Removing a non-existing item from the Large TreeSet Hash Map : Removed key's value:null
Time pass : 0.0332 ms
 Size after removal: 9522

Reaching an existing key's value: x
Time pass : 0.0175 ms

Reaching a non-existing key's value: null
Time pass : 0.0239 ms

Adding a key that won't collide: Size before adding : 9522
Time pass : 0.0047 ms
Size after adding : 9523

Adding a key that will collide: Size before adding : 9523
Time pass : 0.0021 ms
Size after adding : 9524

Adding an existing key: Size before adding : 9524 Value of the key: x
Time pass : 7.0E-4 ms
Size after adding: 9524 Value of the key: t
```



```
Small Coalesced Hash
Adding a key-value pair to the table
Time pass : 0.0014 ms

Size of the Small Coalesced Hash Map : 9
Time pass : 0.0471 ms

Removing an existing item from the Small Coalesced Hash Map : Removed key's value:z
Time pass : 0.0333 ms
 Size after removal: 8

Removing a non-existing item from the Small Coalesced Hash Map : Removed key's value:null
Time pass : 0.0206 ms
 Size after removal: 8

Reaching an existing key's value: x
Time pass : 0.0386 ms

Reaching a non-existing key's value: null
Time pass : 8.4578 ms

Adding a key that won't collide: Size before adding : 8
Time pass : 0.0149 ms
Size after adding : 9

Adding a key that will collide: Size before adding : 9
Time pass : 0.0174 ms
Size after adding : 10

Adding an existing key: Size before adding : 10 Value of the key: x
Time pass : 0.0049 ms
Size after adding: 10 Value of the key: t
```

```
Medium Coalesced Hash
Adding a key-value pair to the table
Time pass : 0.0017 ms

Size of the Medium Coalesced Hash Map : 96
Time pass : 0.0329 ms

Removing an existing item from the Medium Coalesced Hash Map : Removed key's value:z
Time pass : 0.022 ms
 Size after removal: 95

Removing a non-existing item from the Medium Coalesced Hash Map : Removed key's value:null
Time pass : 0.024 ms
 Size after removal: 95

Reaching an existing key's value: x
Time pass : 0.0236 ms

Reaching a non-existing key's value: null
Time pass : 0.0158 ms
Adding a key that won't collide: Size before adding : 95

Time pass : 0.0017 ms
Size after adding : 96

Adding a key that will collide: Size before adding : 96
Time pass : 0.0013 ms
Size after adding : 97

Adding an existing key: Size before adding : 97 Value of the key: x
Time pass : 8.0E-4 ms
Size after adding: 97 Value of the key: t
```



```
Large Coalesced Hash
Adding a key-value pair to the table
Time pass : 0.001 ms

Size of the Large Coalesced Hash Map : 94972
Time pass : 0.0235 ms

Removing an existing item from the Large Coalesced Hash Map : Removed key's value:z
Time pass : 0.0179 ms
 Size after removal: 94971

Removing a non-existing item from the Large Coalesced Hash Map : Removed key's value:null
Time pass : 0.0156 ms
 Size after removal: 94971

Reaching an existing key's value: x
Time pass : 0.0168 ms

Reaching a non-existing key's value: null
Time pass : 0.0163 ms

Adding a key that won't collide: Size before adding : 94971
Time pass : 5.0E-4 ms
Size after adding : 94972

Adding a key that will collide: Size before adding : 94972
Time pass : 0.001 ms
Size after adding : 94973

Adding an existing key: Size before adding : 94973 Value of the key: x
Time pass : 8.0E-4 ms
Size after adding: 94973 Value of the key: t
```