CSE 222/505

DATA STRUCTURES

HOMEWORK 6

REPORT

1801042651

MERVE HORUZ

1) Detailed system requirements

1.1) Non-functional requirements:

openjdk 17.0.1 2021-10-19
OpenJDK Runtime Environment (build 17.0.1+12-Ubuntu-120.04)
OpenJDK 64-Bit Server VM (build 17.0.1+12-Ubuntu-120.04, mixed mode, sharing)

1.2) Functional requirements:

public V get(Object key):

This method takes an object that key and returns value of that key. Key must be exist in hash table.

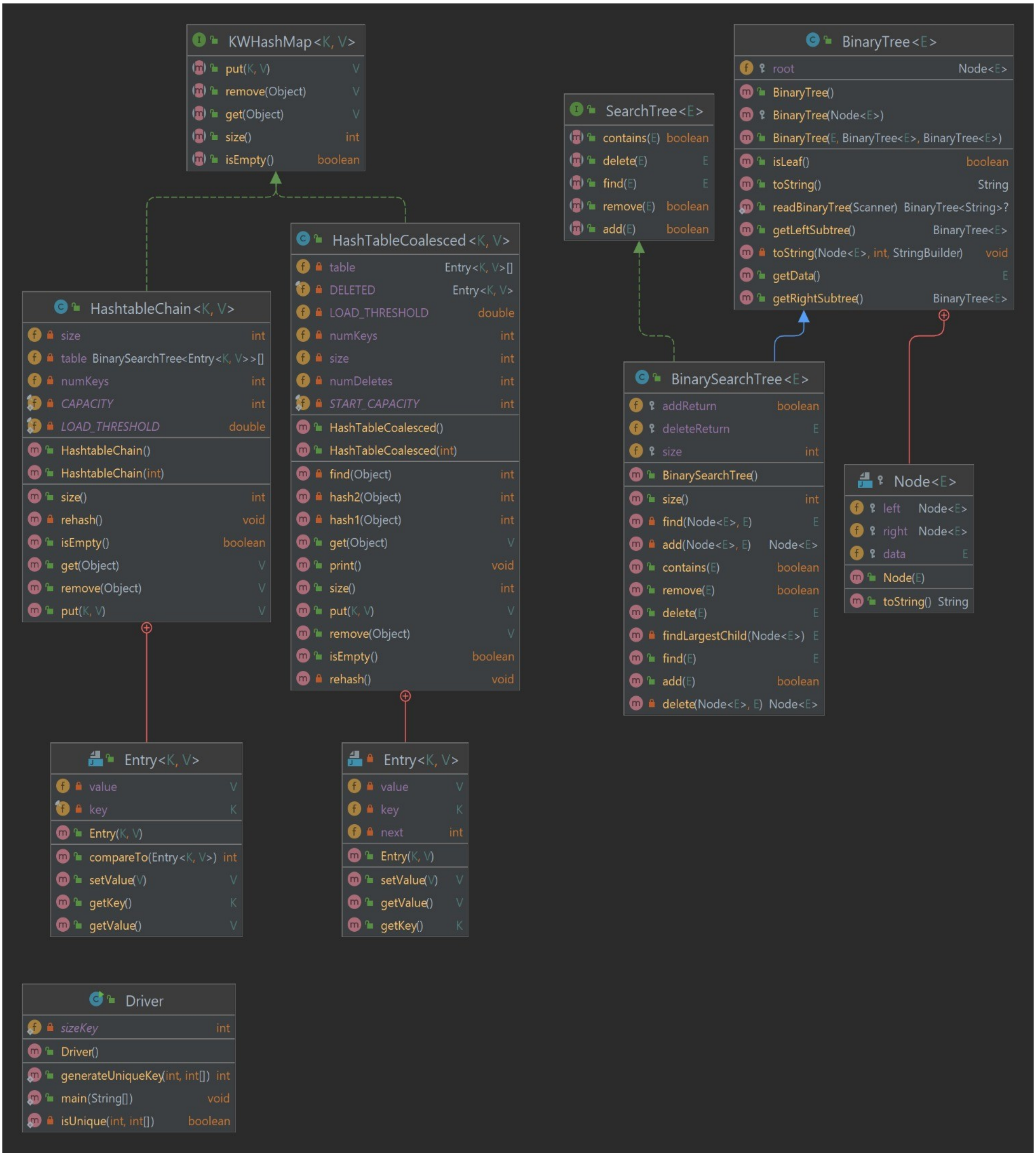
public V put(K key, V value):

This method takes key and value pair. This pair should not be in hash table.

public V remove(Object key):

This method takes an object that key and removes value associated with this key. Key must be exist in hash table.

2) Class diagram



3) Problem solution approach

Hash table with chaining:

First, an array containing the binary search tree is created. Then, the index was determined according to the keys with the help of the hashcode function. If an index is found for more than one value, this value is added to the binary tree in the same index. When a key is to be deleted, the binary search tree is deleted after the index of the value is found.

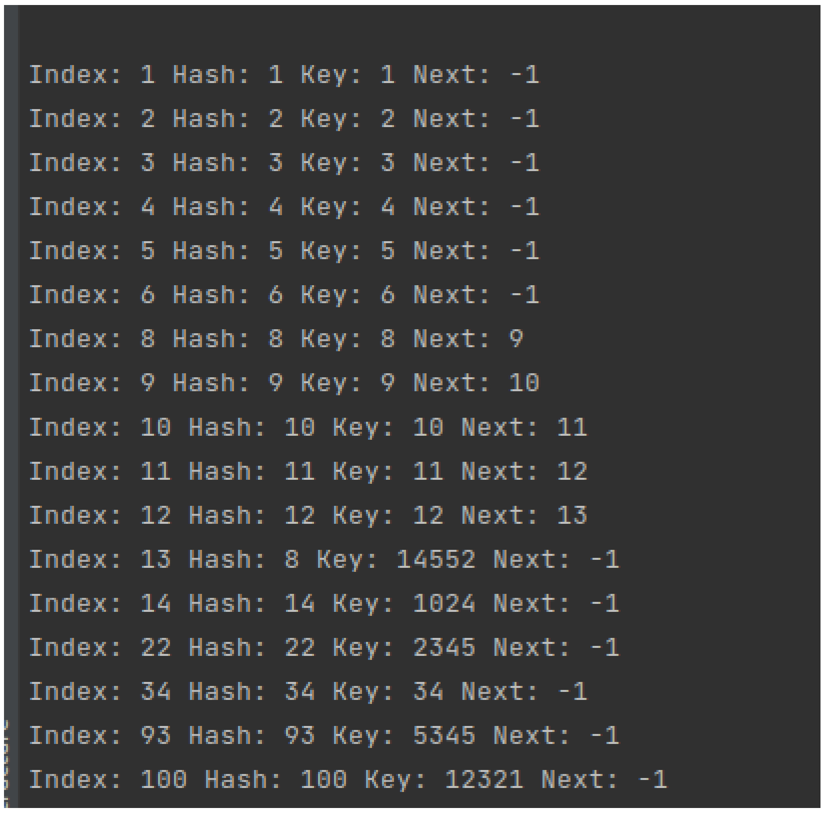
Hash table with coalesced:

Coalesced hashing used for chaining. In addition to the entry class, the next data field has been added to chain the values. If a collision occurs in an index, the double hashing method is used and the next index of this index is pointed generated index by the double hashing method.

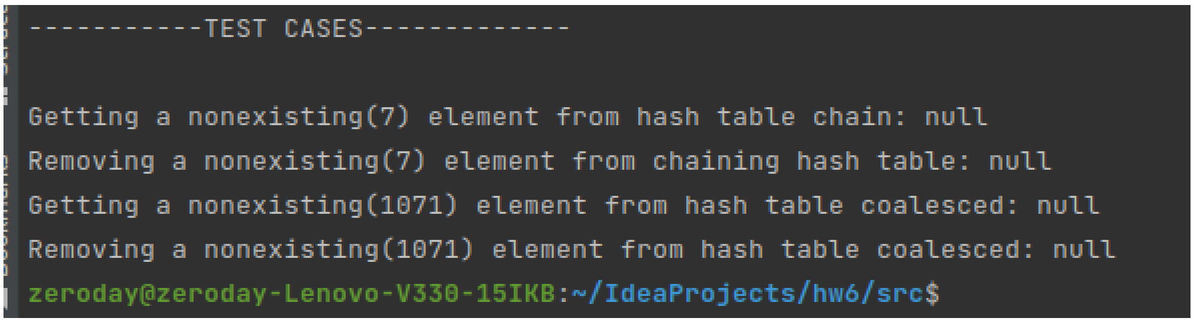
Hashing function:

Hash1 = key % tablesize (10 in our case)
Hash2 = Prime_number – (key % Prime_number)
Hash function = (Hash1 + (i * Hash2)) % tablesize

4) Test cases



```
Index: 1 Hash: 1 Key: 1 Next: -1
Index: 2 Hash: 2 Key: 2 Next: -1
Index: 3 Hash: 3 Key: 3 Next: -1
Index: 4 Hash: 4 Key: 4 Next: -1
Index: 5 Hash: 5 Key: 5 Next: -1
Index: 6 Hash: 6 Key: 6 Next: -1
Index: 8 Hash: 8 Key: 8 Next: 9
Index: 9 Hash: 9 Key: 9 Next: 10
Index: 10 Hash: 10 Key: 10 Next: 11
Index: 11 Hash: 11 Key: 11 Next: 12
Index: 12 Hash: 12 Key: 12 Next: 13
Index: 13 Hash: 8 Key: 14552 Next: -1
Index: 14 Hash: 14 Key: 1024 Next: -1
Index: 22 Hash: 22 Key: 2345 Next: -1
Index: 34 Hash: 34 Key: 34 Next: -1
Index: 93 Hash: 93 Key: 5345 Next: -1
Index: 100 Hash: 100 Key: 12321 Next: -1
```

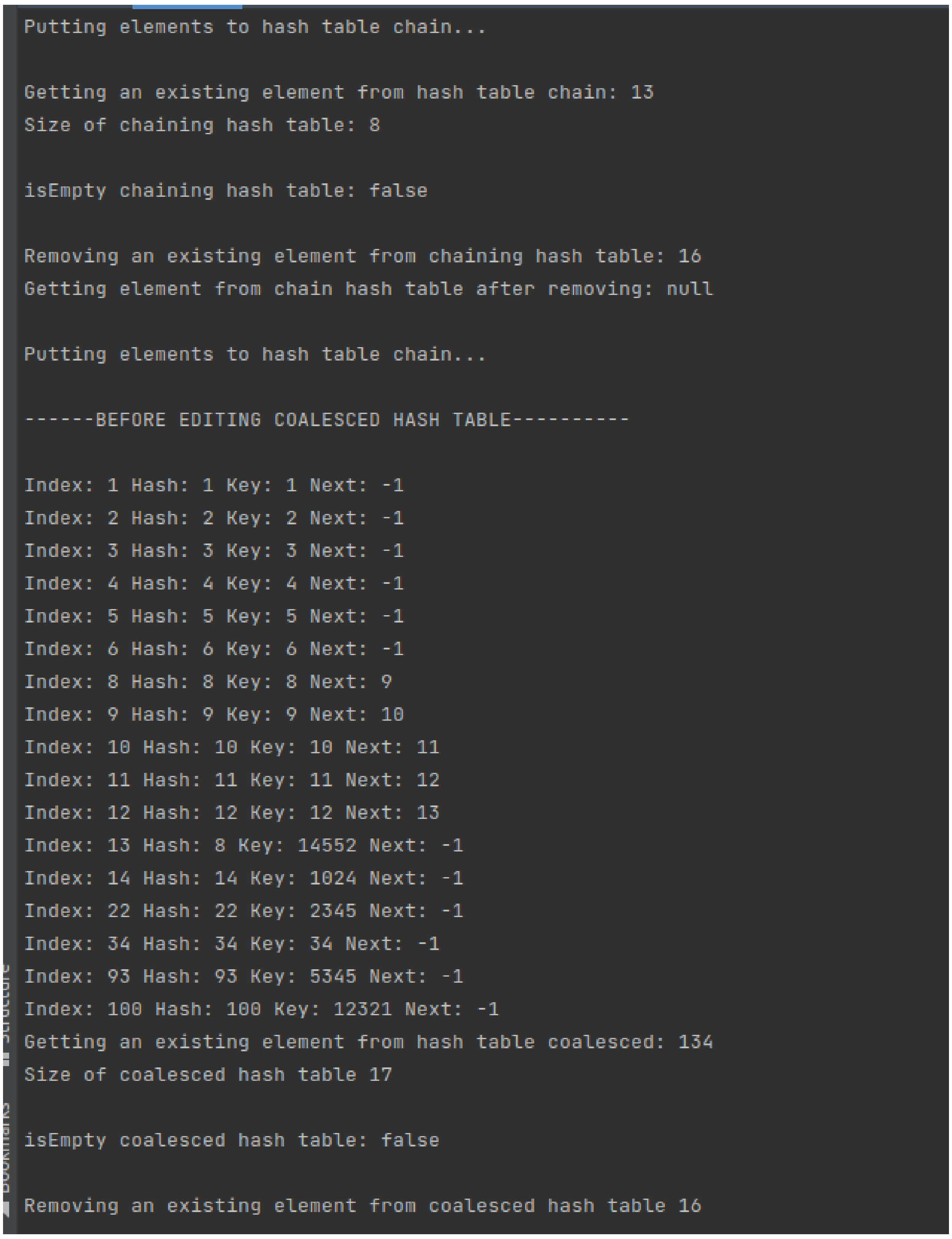


```
-----------TEST CASES-------------

Getting a nonexisting(7) element from hash table chain: null
Removing a nonexisting(7) element from chaining hash table: null
Getting a nonexisting(1071) element from hash table coalesced: null
Removing a nonexisting(1071) element from hash table coalesced: null
zeroday@zeroday-Lenovo-V330-15IKB:~/IdeaProjects/hw6/src$
```

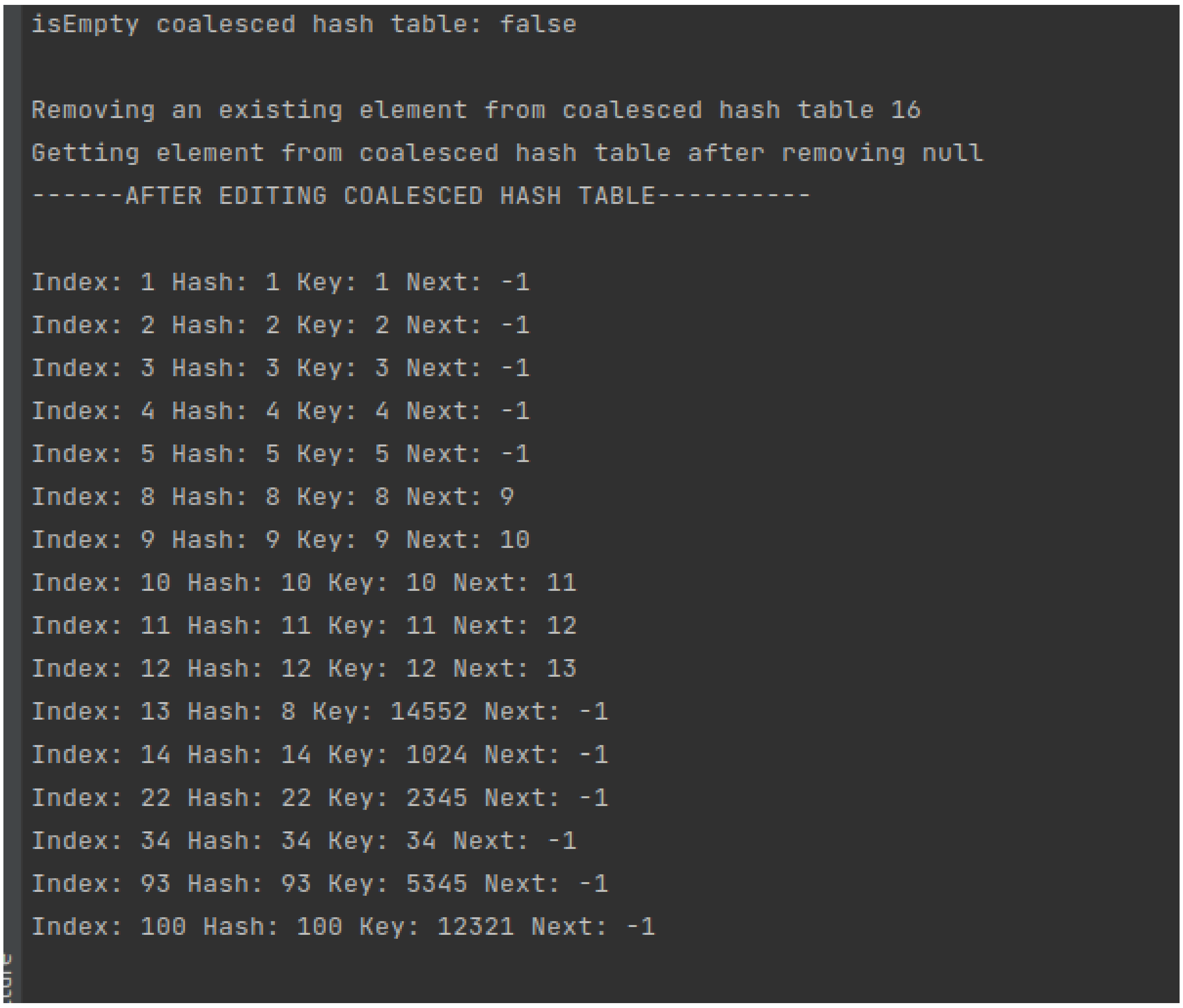
5) Running commands

- make
- java Driver.java
- rm *.class

6) Running results

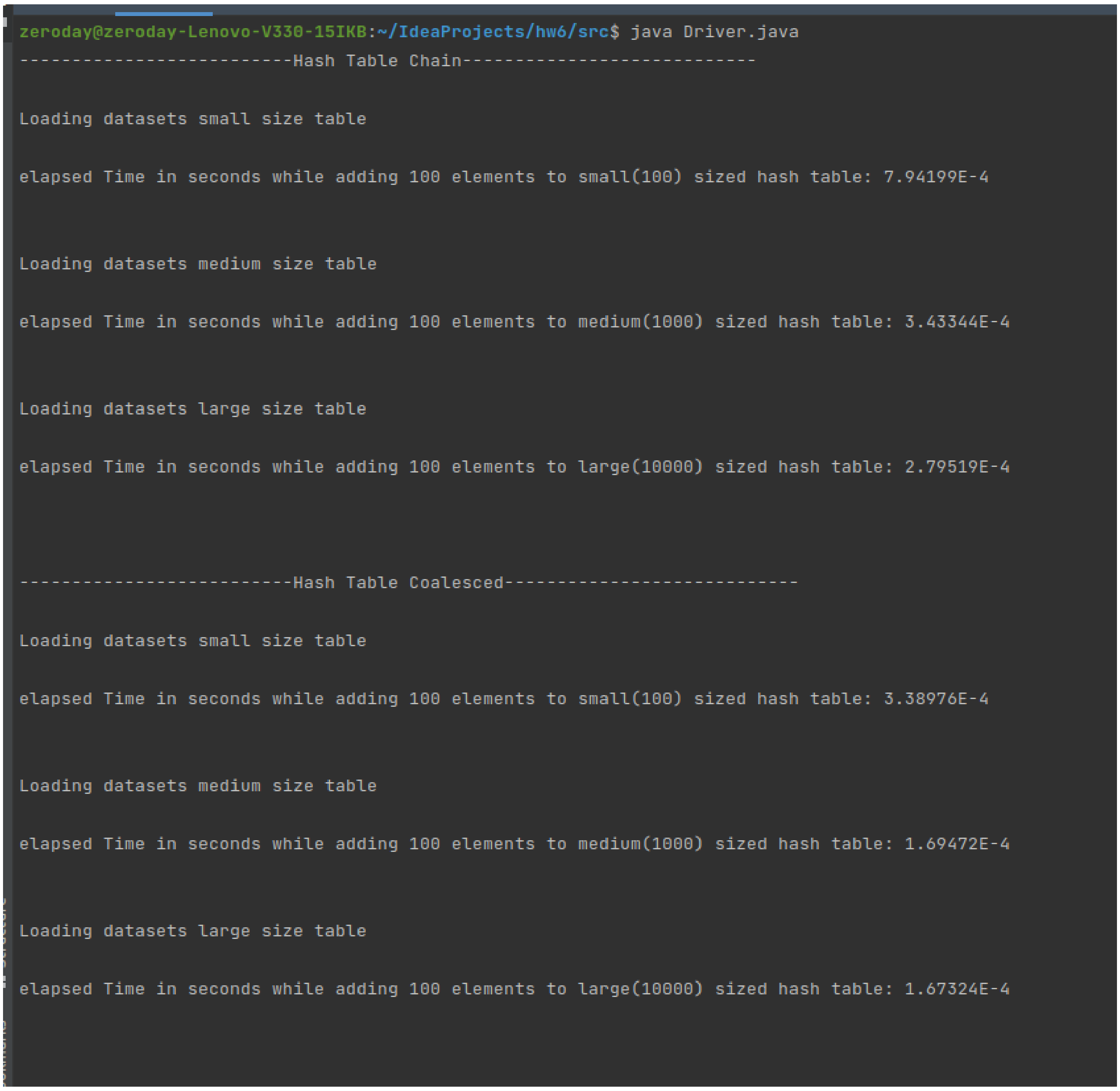


```
Putting elements to hash table chain...

Getting an existing element from hash table chain: 13
Size of chaining hash table: 8

isEmpty chaining hash table: false

Removing an existing element from chaining hash table: 16
Getting element from chain hash table after removing: null

Putting elements to hash table chain...

------BEFORE EDITING COALESCED HASH TABLE----------

Index: 1 Hash: 1 Key: 1 Next: -1
Index: 2 Hash: 2 Key: 2 Next: -1
Index: 3 Hash: 3 Key: 3 Next: -1
Index: 4 Hash: 4 Key: 4 Next: -1
Index: 5 Hash: 5 Key: 5 Next: -1
Index: 6 Hash: 6 Key: 6 Next: -1
Index: 8 Hash: 8 Key: 8 Next: 9
Index: 9 Hash: 9 Key: 9 Next: 10
Index: 10 Hash: 10 Key: 10 Next: 11
Index: 11 Hash: 11 Key: 11 Next: 12
Index: 12 Hash: 12 Key: 12 Next: 13
Index: 13 Hash: 8 Key: 14552 Next: -1
Index: 14 Hash: 14 Key: 1024 Next: -1
Index: 22 Hash: 22 Key: 2345 Next: -1
Index: 34 Hash: 34 Key: 34 Next: -1
Index: 93 Hash: 93 Key: 5345 Next: -1
Index: 100 Hash: 100 Key: 12321 Next: -1
Getting an existing element from hash table coalesced: 134
Size of coalesced hash table 17

isEmpty coalesced hash table: false

Removing an existing element from coalesced hash table 16
```



```
isEmpty coalesced hash table: false

Removing an existing element from coalesced hash table 16
Getting element from coalesced hash table after removing null
------AFTER EDITING COALESCED HASH TABLE----------

Index: 1 Hash: 1 Key: 1 Next: -1
Index: 2 Hash: 2 Key: 2 Next: -1
Index: 3 Hash: 3 Key: 3 Next: -1
Index: 4 Hash: 4 Key: 4 Next: -1
Index: 5 Hash: 5 Key: 5 Next: -1
Index: 8 Hash: 8 Key: 8 Next: 9
Index: 9 Hash: 9 Key: 9 Next: 10
Index: 10 Hash: 10 Key: 10 Next: 11
Index: 11 Hash: 11 Key: 11 Next: 12
Index: 12 Hash: 12 Key: 12 Next: 13
Index: 13 Hash: 8 Key: 14552 Next: -1
Index: 14 Hash: 14 Key: 1024 Next: -1
Index: 22 Hash: 22 Key: 2345 Next: -1
Index: 34 Hash: 34 Key: 34 Next: -1
Index: 93 Hash: 93 Key: 5345 Next: -1
Index: 100 Hash: 100 Key: 12321 Next: -1
```

Q1-3)



```
zeroday@zeroday-Lenovo-V330-15IKB:~/IdeaProjects/hw6/src$ java Driver.java
--------------------------Hash Table Chain----------------------------

Loading datasets small size table

elapsed Time in seconds while adding 100 elements to small(100) sized hash table: 7.94199E-4


Loading datasets medium size table

elapsed Time in seconds while adding 100 elements to medium(1000) sized hash table: 3.43344E-4


Loading datasets large size table

elapsed Time in seconds while adding 100 elements to large(10000) sized hash table: 2.79519E-4



--------------------------Hash Table Coalesced----------------------------

Loading datasets small size table

elapsed Time in seconds while adding 100 elements to small(100) sized hash table: 3.38976E-4


Loading datasets medium size table

elapsed Time in seconds while adding 100 elements to medium(1000) sized hash table: 1.69472E-4


Loading datasets large size table

elapsed Time in seconds while adding 100 elements to large(10000) sized hash table: 1.67324E-4
```

Q1-2-a)

Coalesced hashing is a technique for implementing a hash table. It's an open addressing technique which means that all keys are stored in the array itself As opposed to other open addressing techniques however, it also uses nodes with next-poiners to form collision chains.

This strategy is effective, efficient, and very easy to implement. However, sometimes the extra memory use might be prohibitive, and the most common alternative, open addressing, has uncomfortable disadvantages that decrease performance. The primary disadvantage of open addressing is primary and secondary clustering, in which searches may access long sequences of used buckets that contain items with different hash addresses; items with one hash address can thus lengthen searches for items with other hash addresses.

b)

Double hashing is a computer programming technique used in conjunction with open addressing in hash tables to resolve hash collisions, by using a secondary hash of the key as an offset when a collision occurs.

Advantage:

- Double hashing finally overcomes the problems of the clustering issue.

Disadvantages:

- Double hashing is more difficult to implement than any other.
- Double hashing can cause thrashing.