Hashing Code

Ex. 1

```
package A8HashSet;
import java.util.HashSet; //Need to import HashSet
import java.util.Iterator; //Need to import Iterator for traversing on set
public class A1HashSet {

	public static void main(String[] args) {
		//Creation
		HashSet<Integer> set = new HashSet<Integer>();
	  //ArrayList<Integer> list = new ArrayList<Integer>();
    //HashSet is similar with ArrayList Syntax

		//Insert
		set.add(1); //similar syntax with list.add(1);
		set.add(2);
		set.add(3);
		set.add(1); //It will not take dublicate so value in set remains 1,2,3 only

		//size
		 System.out.println("Size of set: "+set.size()); //Ans:3

		//Print all elements in the set
		 System.out.println(set); //Ans: [1, 2, 3]

		//Iterator     for traversing on each element of set like i in for loop traverse on array(i is iterator for array)
```

Hashing Code

```
                //and need to import Iterator
            Iterator it = set.iterator(); //set has iterator method which returns iterator for set
                        //here 'it' is iterator like i is iterator for traversing array and Iterator is
type of 'it'
            //hasNext and next are two special function of 'it'
            while(it.hasNext()) {
                System.out.println(it.next());  //Ans: 1 2 3   here order/sequence may vary
                  //no need to write it++ etc
            }

            //Search    here in set for search special function is used that is "contains"
            if(set.contains(1)) {
                System.out.println("set contains 1 "); //Ans: set contains 1
            }
            if(!set.contains(6)) {
                System.out.println("does not contain 6"); //Ans: does not contain 6
            }

            //Delete
            set.remove(1); //here 1 is an element and not an index
            if(!set.contains(1)) {
                System.out.println("delete the 1"); //Ans:delete the 1
            }
            System.out.println(set); //Ans:[2, 3]
        }
```

Hashing Code

}

Ex.2

```
package A9HashMap;

import java.util.HashMap; //need to import HashMap

import java.util.Map;

import java.util.Set;

public class A1HashMap {


	public static void main(String[] args) {

		//Creation of HashMap  Syntax similar with HashSet

		//country(key),population(value)

		HashMap<String,Integer> map = new HashMap<>();  //here key is String and value is
Integer


		//Insertion

		map.put("India", 120);  //put method used for insertion

		map.put("China", 130);

		map.put("US", 70);


		System.out.println(map); //{China=130, US=70, India=120}

		//(map are unordered set. they can print value in any sequence)


		map.put("China", 180);    //If key is already present then it update the value of that key

		System.out.println(map);//{China=180, US=70, India=120} here China value becomes
180
```

Hashing Code

```
		//Search
		if(map.containsKey("China")) {  //check key is present or not and returns true or false according to that
			System.out.println("Key is present in a map");  //Ans: Key is present in a map
		}else {
			System.out.println("Key is not present in the map");
		}

		System.out.println(map.get("China")); //key exists //Ans:180
		System.out.println(map.get("Indonesia")); //key doesn't exist //Ans: null

		int arr[]= {12,15,18};
		//regular way to iterate array and  print values
		for(int i=0;i<arr.length;i++) {
			System.out.print(arr[i]+" "); //Ans:12 15 18
		}
		System.out.println();
		//new way to iterate array and print them
		for(int val: arr) {
			System.out.print(val+" "); //Ans:12 15 18
		}
		System.out.println();
		//for iteration on map
		//similar with for(int val: arr) here int=Map.Entry<String, Integer> and val=e and arr=map.entrySet()
```

Hashing Code

```
		for(Map.Entry<String, Integer> e: map.entrySet()) { //with help of entrySet
			System.out.print(e.getKey()+" ");
			System.out.println(e.getValue());  /* Ans

			*China 180
	US 70
	India 120
			*/

		}

		Set<String> keys = map.keySet(); //with the help of keySet()
		for(String key: keys) {
			System.out.println(key+" "+map.get(key));
	/* Ans
			 *China 180
	  US 70
	  India 120
			*/
		}

		System.out.println(map);//Ans: {China=180, US=70, India=120}
		//remove
		map.remove("US");
		System.out.println(map);//Ans: {China=180, India=120}
```

Hashing Code

```
		//size of map

		System.out.println("size: "+map.size());

	}


}
```

Ex.3

```
package B1HashMapImplementation;
import java.util.*;
public class A1HashMapImplementation {
	static class HashMap<K,V>{ //Generics
		private class Node{
			K key;			//K and V are datatype
			V value;
			public  Node(K key, V value){
				this.key=key;
				this.value=value;
			}
		}

		private int n; //total no of nodes
		private int N; //total no of buckets or array length
		//private int arr[];
		private LinkedList<Node> buckets[]; //N=buckets.length

//		@SuppressWarnings("unchecked")
		public HashMap() { //constructor of HashMap
			this.N=4;
			this.buckets= new LinkedList[4];
			for(int i=0;i<4;i++) {
				this.buckets[i] = new LinkedList<>(); //here we create
empty LinkedList at each index of array so later we can add node
			}
		}

		private int hashFunction(K key) {
			int bi = key.hashCode(); //this is buld-in method in java which
gives bucket index
			//but return value may be negative or positive and sometime
greater than size of bucket that is >N
			//we need bi index between 0 to (N-1) so we take remainder of
bi
			return Math.abs(bi)%N; //Math.abs for [positive value and %N
for remainder
		}
		private int searchInLL(K key,int bi) {
```

Hashing Code

```
                        LinkedList<Node> ll = buckets[bi]; //here we take LinkedList at
that bi
                        for(int i=0;i<ll.size();i++) {
                                if(ll.get(i).key == key) {
                                        return i; //here i means di
                                }
                        }
                        return -1; //if key not found then send -1
                }

                private void rehash() {
                        LinkedList<Node> OldBuckets[] = buckets;
                        buckets = new LinkedList[N*2];
                        for(int i=0;i<N*2;i++) {
                                buckets[i] = new LinkedList<>(); //here we create empty
linkedList at each array index
                        }
                        for(int i=0;i<OldBuckets.length;i++) {
                                LinkedList<Node> ll = OldBuckets[i];
                                for(int j=0;j<ll.size();j++) {
                                        Node node = ll.get(i);
                                        put(node.key,node.value);
                                }
                        }
                }
                public void put(K key, V value) {
                        int bi = hashFunction(key); //here bi means bucket index
                        int di = searchInLL(key,bi); //here di means data index and LL
in searchInLL means LinkedList
                         //if di=-1 means key is not exist and if di>=0 means key is
exist
                        if(di == -1) { //key doesn't exist
                                buckets[bi].add(new Node(key,value));
                                n++; //no of node increases
                        }else { //key is exist
                             Node node =  buckets[bi].get(di); //we take node of di
where node has key and value and we update the value;
                             node.value = value;
                        }

                        double lambda = (double)n/N;
                        if(lambda > 2.0) {
                                //reHashing
                                rehash();
                        }
                }

                public V get(K key) {
                        int bi = hashFunction(key); //here bi means bucket index
                        int di = searchInLL(key,bi); //here di means data index and LL
in searchInLL means LinkedList
                         //if di=-1 means key is not exist and if di>=0 means key is
exist
                        if(di == -1) { //key doesn't exist
                             return null;
```

Hashing Code

```
                    }else { //key is exist
                        Node node =  buckets[bi].get(di); //we take node of di
where node has key and value and we update the value;
                        return node.value;
                    }
                }


                public boolean containsKey(K key) {
                    int bi = hashFunction(key); //here bi means bucket index
                    int di = searchInLL(key,bi); //here di means data index and LL
in searchInLL means LinkedList
                     //if di=-1 means key is not exist and if di>=0 means key is
exist
                    if(di == -1) { //key doesn't exist
                        return false;
                    }else { //key is exist
                        return true;
                    }
                }

                public V remove(K key) {
                    int bi = hashFunction(key); //here bi means bucket index
                    int di = searchInLL(key,bi); //here di means data index and LL
in searchInLL means LinkedList
                     //if di=-1 means key is not exist and if di>=0 means key is
exist
                    if(di == -1) { //key doesn't exist
                          return null;
                    }else { //key is exist
                        Node node =  buckets[bi].remove(di); //we take node of di
where node has key and value
                        n--;
                        return node.value;
                    }

                }

                public boolean isEmpty() {
                    return n==0;
                }

                public ArrayList<K> keySet(){
                    ArrayList<K> keys = new ArrayList<K>();
                    for(int i=0;i<buckets.length;i++) {  //i means bucket index
                        LinkedList<Node> ll = buckets[i];
                        for(int j=0;j<ll.size();j++) {
                            Node node = ll.get(j);
                            keys.add(node.key);
                        }
                    }
                    return keys;
                }
        }
```

Hashing Code

```
	public static void main(String[] args) {
		HashMap<String,Integer> map = new HashMap<>();  //Here we use the custom HashMap class and not build-in HashMap in Java
		map.put("India", 190);
		map.put("China", 200);
		map.put("US", 50);

		ArrayList<String> keys = map.keySet();
		for(int i=0;i<keys.size();i++) {
			System.out.println(keys.get(i)+" "+map.get(keys.get(i)));
		}

		System.out.println(map.remove("India"));

	}

}
```

Ex.4

```
package B2HashingQuestion;
import java.util.*;
//Que. Given an integer array of size n, find all elements that appear more than [n/3] times.
//e.g int nums[] = {1,3,2,5,1,3,1,5,1}; and ans= 1 because 1 appear more than n/3=9/3=3
//Note: if you solve with BruteForce method(nested loops) then it takes O(n^2) time complexity
// and if we use one loop with HashMap then it takes O(n) time complexity
public class A1MajorityElement {

	public static void main(String[] args) {
		int nums[] = {1,3,2,5,1,3,1,5,1};
		HashMap<Integer,Integer> map = new HashMap<>();
		for(int ele:nums) {  //time complexity: O(n)
			if(map.containsKey(ele)) {
				map.put(ele, map.get(ele)+1);
			}else {
				map.put(ele, 1);
			}
		}

		for(int key: map.keySet()) { //time complexity: O(n)
			if(map.get(key)>(nums.length/3)) {
				System.out.println(key);	//Ans: 1
			}
		}
	}

}
```

Ex.5

```
package B2HashingQuestion;
import java.util.*;
//Que. Union of two arrays
//arr1[]={7,3,9}; and arr2[]={6,3,9,2,9,4};   and union of these two array is
Union[]={7,3,9,6,2,4};
//Note: If we solve this question with BruteForce(Using nested loops then it takes
O(n^2) complexity and if we sort the arrays and then solve
// it takes O(nlogn) time complexity and If solve using HashSet then it takes O(n)
time complexity
//Here we solve using HashSet
public class A2UnionOfTwoArrays {
    public static void unionOfTwoArrays(int arr1[],int arr2[]) {
       HashSet<Integer> set = new HashSet<>();
       for(int ele:arr1) { // time complexity:O(n);
              set.add(ele);
       }
       for(int ele:arr2) { // time complexity:O(n);
              set.add(ele);
       }
       System.out.println(set.size()); //Ans: 6
       System.out.println(set); //Ans: [2, 3, 4, 6, 7, 9]

       Iterator it = set.iterator();
       while(it.hasNext()) {
              System.out.print(it.next()+" ");//Ans:2 3 4 6 7 9
       }
    }
       public static void main(String[] args) {
              int arr1[]={7,3,9};
              int arr2[]={6,3,9,2,9,4};
              unionOfTwoArrays(arr1,arr2);
       }

}
```

Ex.6

```
package B2HashingQuestion;
import java.util.*;
//Que. find the intersection of two arrays that is common value between two arrays
//arr1[]={7,3,9}; and arr2[]={6,3,9,2,9,4}; and ans should be 3,9 and count=2;
public class A3InterSectionOfTwoArrays {
//   IMP Note: To check Occurrence/frequency/common/Union/Unique  element in
Array/ArrayList/LinkedList we use Hashing(HashSet,HashMap);
       public static int intersection(int arr1[], int arr2[]) {
              int count=0;
                //for making unique value arr we convert 1st array into set
              HashSet<Integer> setArr1 = new HashSet<>();
              for(int ele: arr1) {
                     setArr1.add(ele);
              }
```

```
		for(int i=0;i<arr2.length;i++) {
			if(setArr1.contains(arr2[i])) {
				count++;
				System.out.print(arr2[i]+" "); //Ans: 3 9
				setArr1.remove(arr2[i]); //here we remove from the first
set Array
			}
		}

		return count;
	}

	public static void main(String[] args) {
		int arr1[] = {7,3,9};
		int arr2[] = {6,3,9,2,9,4};

		int commonVal = intersection(arr1,arr2);
	 System.out.println();
	 System.out.println("Count of value inside intersection of two arrays:
"+commonVal); //Ans: 2
	}

}
```

Ex.7

```
package B2HashingQuestion;
import java.util.*;
//Que. Find Itinerary from tickets here itinerary means journey/path  //refer diagram
for more understanding
//Condition: No loop will create and Not from two location to one destination
// So Keys array/set is unique and value's array/set also unique
public class A4FindItineraryFromTickets {

	public static void itinerary(HashMap<String,String> tickets) {
		String start="";
		HashMap<String, String> revMap = new HashMap<>(); //here revMap used
just for finding the start
		for(String key: tickets.keySet()) {
			revMap.put(tickets.get(key), key);
		}
		for(String key:tickets.keySet()) {
			if(revMap.containsKey(key)==false) {
				start= key;
				break;
			}
		}
		while(tickets.containsKey(start)) {
			System.out.print(start+"-> "); //Ans : Mumbai-> Delhi-> Goa->
Chennai->
			start = tickets.get(start);
```

```
			}
			System.out.println(start); //this is for last destination Ans:
Bengaluru
		}

		public static void main(String[] args) {
			HashMap<String,String> tickets = new HashMap<>();
			tickets.put("Chennai", "Bengaluru");
			tickets.put("Mumbai", "Delhi");
			tickets.put("Goa", "Chennai");
			tickets.put("Delhi", "Goa");

			itinerary(tickets);  //Ans: Mumbai-> Delhi-> Goa-> Chennai-> Bengaluru
		}

}
```

Ex.8

```
package B2HashingQuestion;
import java.util.*;
public class A5NoOfSubArrays {
  //Que. Count no of subArrays who's sum equal to k
	//e.g int arr[]={10,2,-2,-20,10} k=-10;  Ans should be 3 subArrays
	public static void main(String[] args) {
		int arr[]= {10,2,-2,-20,10};
		int k=-10;
		HashMap<Integer,Integer> map = new HashMap<>(); //<sum,frequency>

		map.put(0, 1); //empty sub array comes 1 time So frequency 1 and sum=0;
		int ans=0;
		int sum=0;
		for(int j=0;j<arr.length;j++) {
			sum += arr[j];    //later we add sum in hashmap for each
iteration
			if(map.containsKey(sum-k)) {
				ans += map.get(sum-k);
			}

			if(map.containsKey(sum)) {
				map.put(sum, map.get(sum)+1);
			}else {
				map.put(sum,1);
			}
		}
		System.out.println("Ans: "+ans); //Ans : 3
	}

}
```