#HASHING

Code for hash operations:

```
import java.util.*;
public class hashoperations {

    public static void main(String[] args) {
        HashMap<String , Integer> hm= new HashMap<>();
        // hm operations like get put containskey size remove isEmpty
        hm.put("A",1);
        hm.put("B",2);
        hm.put("C", 3);
        hm.put("D",4);

        System.out.print(hm);
        System.out.println();

        int val= hm.get("A");
        System.out.println(val);

        System.out.println(hm.containsKey("i"));

        System.out.println(hm.size());

        System.out.println(hm.remove("A"));

        //clear function clears the data in hashmap
        hm.clear();
        System.out.println(hm.isEmpty());
    }

}
Output:
{A=1, B=2, C=3, D=4}
1
false
4
1
true
```

Code for iteration loop in hashing:

```java
import java.util.*;


public class iterationinHashes {
    public static void main(String[] args) {
        HashMap<Integer,Integer> hm= new HashMap<>();
        hm.put(1,10);
        hm.put(2,20);
        hm.put(3,30);
        hm.put(4,40);

        Set<Integer> key = hm.keySet();
        System.out.println(key);
        // printing k, v in pairs we use foreach loop
        for(Integer i:key){
            System.out.print(i);
        }

    }
}
```

```
[1, 2, 3, 4]
1234
```

Linked Hash Map:

```java
import java.util.*;
class LinkedHashMaps{
    public static void main(String[] args) {
        LinkedHashMap<String,Integer> lhm= new LinkedHashMap();
        lhm.put("a",100);
        lhm.put("b",34);
        lhm.put("c",300);
        lhm.put("d",50);

       HashMap<String,Integer> hm= new HashMap();
        hm.put("a",100);
        hm.put("b",34);
        hm.put("c",300);
        hm.put("d",5);

      TreeMap<String,Integer> tm= new TreeMap();
        tm.put("aa ",100);
        tm.put("aa`",34);
```

```
        tm.put("aaa",300);
        tm.put("a",5);

    System.out.println(lhm);
    System.out.println(hm);
    System.out.println(tm);;

    }

}
Output:
{a=100, b=34, c=300, d=5}
{a=100, b=34, c=300, d=5}
{a=5, aa =100, aa`=34, aaa=300}
```

Code For valid anagram or not :

```
import java.util.*;
public class isAnagram {
    public static boolean isAnagram(String s,String t){
        HashMap<Character , Integer > map= new HashMap<>();

        for(int i=0;i<s.length();i++){
            char ch= s.charAt(i);
            map.put(s.charAt(i),map.getOrDefault(s.charAt(i), 0)+1);

        }

        for(int i=0;i<t.length();i++){
            char ch= t.charAt(i);
            if(map.get(ch)!=null){
                if(map.get(ch)!=null){
                    if(map.get(ch)==1){
                        map.remove(ch);

                    }else{
                        map.put(ch,map.get(ch)-1);
                    }
                }
                else{
                    return false;
                }
```

```
            }
        } return map.isEmpty();
    }
    public static void main(String[] args) {
        String s= "race";
        String t="care";
        System.out.println(isAnagram(s, t));
    }

}
Output:
true
```

HashSet Operations:

```
import java.util.*;
import java.util.HashSet;
public class hashSet {
    public static void main(String args[]){
        HashSet <Integer> hs= new HashSet<>();
        hs.add(1);
        hs.add(2);
        hs.add(3);
        hs.add(4);
        hs.add(2);
        System.out.println(hs);
        System.out.println(hs.contains(2));
        hs.remove(2);
        System.out.println(hs.contains(2));
        System.out.println(hs.size());
        hs.clear();
        System.out.println(hs);
    }
}
Output:
[1, 2, 3, 4]
true
false
3
[]
```

Iterator in HashSets:

```
import java.util.*;
public class iterater {
    public static void main(String args[]){
        HashSet<String > cities= new HashSet<>();
        cities.add("hyderabad");
        cities.add("bengalore");
        cities.add("noida");
        cities.add("vadodara");
        // iterater
        Iterator it = cities.iterator();
        while(it.hasNext()){
            System.out.println(it.next());
        }
System.out.println();
        //enhanced for-each loop

        for(String city: cities){
            System.out.println(city);
        }
    }

}output:
hyderabad
vadodara
noida
bengalore

hyderabad
vadodara
noida
bengalore
```

Linjed Hash Set :

```
import java.util.*;
public class LinkedHashset {
    public static void main(String[] args) {


    LinkedHashSet<Integer>lhs= new LinkedHashSet<>();
    lhs.add(1);
```

```
    lhs.add(3);
    lhs.add(5);
    lhs.add(9);
    System.out.println(lhs);
    System.out.println(lhs.size());
    lhs.remove(5);
    System.out.println(lhs);


}
}output:
[1, 3, 5, 9]
4
[1, 3, 9]
```

Tree set:

```
import java.util.*
;
public class treeset {
    public static void main(String[] args) {
        TreeSet<Integer> ts= new TreeSet<>();
        ts.add(2);
        ts.add(9);
        ts.add(112);
        ts.add(6);
        System.out.println(ts);
        System.out.println(ts.size());
        ts.remove(2);
        System.out.println(ts.size());
    }
}
Output:
[2, 6, 9, 112]
4
3
```

Find Distinct(Unique) elements:

```
import java.util.*;
public class countUniqueElements {
    public static void main(String[] args) {
        HashSet<Integer> hs= new HashSet();
```

```
        int arr[]={1,2,3,4,5,6,7,1,2,3};
        for(int i=0;i<arr.length;i++){
            hs.add(arr[i]);
        }
        System.out.println(hs);
        System.out.println(hs.size());
    }
}
Output:
[1, 2, 3, 4, 5, 6, 7]
7
```

Largest Subarray problem:

```
import java.util.HashMap;
public class largestSubarray {
public static void main(String args[] ){
    int arr[]={5,-2,2,-8,1,7,10,3};
    HashMap<Integer,Integer>map= new HashMap<>();
    int sum=0;
    int len=0;
    for(int j=0;j<arr.length;j++){
        sum+=arr[j];
        if(map.containsKey(sum)){
            len=Math.max(len,j-map.get(sum));
        }else{
            map.put(sum,j);
        }
    }
    System.out.println(len);
}

}
Output:5
```

Subarray sum equals to k:

```
import java.util.HashMap;
public class SubarraysumequaltoK {
public static void main(String args[] ){
    int arr[]={10,2,-2,-20,10};
    int k=-10;
```

```
    HashMap<Integer,Integer>map= new HashMap<>();
    map.put(0,1);
    int sum=0;
    int ans=0;
    for(int j=0;j<arr.length;j++){
        sum+=arr[j];
        if(map.containsKey(sum-k)){
           ans+=map.get(sum-k);
        }
        map.put(sum,map.getOrDefault(sum, 0)+1);
    }
    System.out.println(ans);
}

}
Output:
3
```