**Write a Java program to insert elements at the first and last positions of a linked list.** Import java.util.LinkedList;

Public class LinkedListInsertExample {

Public static void main(String[] args) {

// Create a LinkedList of Strings

LinkedList<String> list = new LinkedList<>();

// Add some initial elements

List.add(“B”);

List.add(“C”);

System.out.println(“Original LinkedList: “ + list);

// Insert at the first position

List.addFirst(“A”);

// Insert at the last position

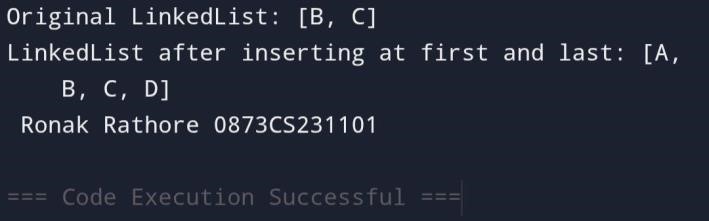
List.addLast(“D”);

// Print the updated LinkedList

System.out.println(“LinkedList after inserting at first and last: “ + list);

System.out.println(“ Ronak Rathore 0873CS231101”);

}}



**Write a Java program to add all elements from one TreeSet to another TreeSet.**

import java.util.TreeSet;

public class TreeSetAddAllExample { public static void main(String[] args) {

// First TreeSet

TreeSet<String> set1 = new TreeSet<>(); set1.add("Apple"); set1.add("Banana"); set1.add("Mango");

// Second TreeSet

TreeSet<String> set2 = new TreeSet<>(); set2.add("Orange"); set2.add("Grapes");

// Display both sets before adding

System.out.println("TreeSet 1: " + set1);

System.out.println("TreeSet 2 (before adding): " + set2);

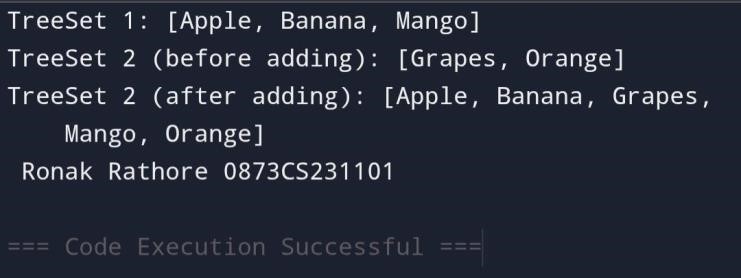
// Add all elements from set1 to set2 set2.addAll(set1);

// Display TreeSet 2 after adding

System.out.println("TreeSet 2 (after adding): " + set2);

System.out.println(“ Ronak Rathore 0873CS231101”);

}}



**Write a Java program to display the elements of a TreeSet in reverse order.**

import java.util.TreeSet; import java.util.Iterator;

public class TreeSetReverseOrder { public static void main(String[] args) {

// Create a TreeSet and add elements TreeSet<String> fruits = new TreeSet<>(); fruits.add("Apple"); fruits.add("Mango"); fruits.add("Banana"); fruits.add("Orange");

System.out.println("TreeSet in ascending order: " + fruits);

System.out.println("TreeSet in reverse order:");

Iterator<String> reverseIterator = fruits.descendingIterator(); while (reverseIterator.hasNext()) {

System.out.println(reverseIterator.next());

}

System.out.println(" Ronak Rathore 0873CS231101");

}}



**Write a Java program to retrieve the first and last elements from a TreeSet.**

Import java.util.TreeSet;

Public class TreeSetFirstLastExample {

Public static void main(String[] args) {

TreeSet<String> animals = new TreeSet<>();

Animals.add(“Zebra”);

Animals.add(“Elephant”);

Animals.add(“Lion”);

Animals.add(“Giraffe”);

System.out.println(“TreeSet: “ + animals);

String first = animals.first();

System.out.println(“First element: “ + first);

String last = animals.last();

System.out.println(“Last element: “ + last);

System.out.println(“ Ronak Rathore 0873CS231101”);

}}



**Write a Java program to clone a TreeSet into another TreeSet**

Import java.util.TreeSet;

Public class TreeSetCloneExample {

Public static void main(String[] args) { TreeSet<String> originalSet = new TreeSet<>(); originalSet.add(“Apple”); originalSet.add(“Banana”); originalSet.add(“Mango”);

System.out.println(“Original TreeSet: “ + originalSet);

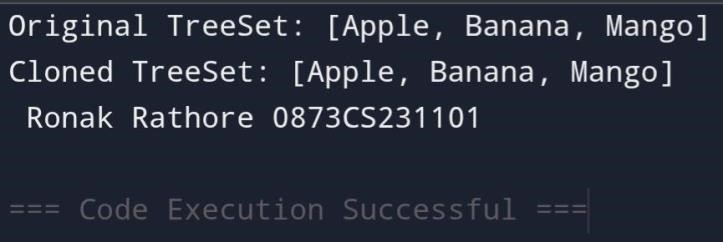
@SuppressWarnings(“unchecked”)

TreeSet<String> clonedSet = (TreeSet<String>) originalSet.clone();

System.out.println(“Cloned TreeSet: “ + clonedSet);

System.out.println(“ Ronak Rathore 0873CS231101”);

}}



**Write a Java program to count the number of elements .**

Import java.util.TreeSet;

Public class TreeSetCount {

Public static void main(String[] args) {

TreeSet<String> colors = new TreeSet<>();

Colors.add(“Red”);

Colors.add(“Green”);

Colors.add(“Blue”);

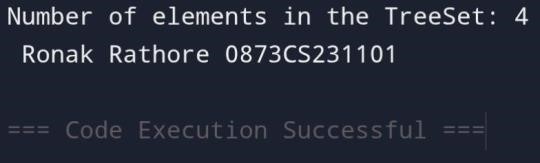
Colors.add(“Yellow”);

System.out.println(“Number of elements in the TreeSet: “ + colors.size());

System.out.println(“ Ronak Rathore 0873CS231101”);

}

}



**Write a Java program to compare two TreeSets.**

Import java.util.TreeSet;

Public class TreeSetCompare {

Public static void main(String[] args) {

TreeSet<String> set1 = new TreeSet<>();

Set1.add(“Apple”);

Set1.add(“Banana”);

Set1.add(“Mango”);

TreeSet<String> set2 = new TreeSet<>();

Set2.add(“Banana”);

Set2.add(“Mango”);

Set2.add(“Apple”);

Boolean isEqual = set1.equals(set2);

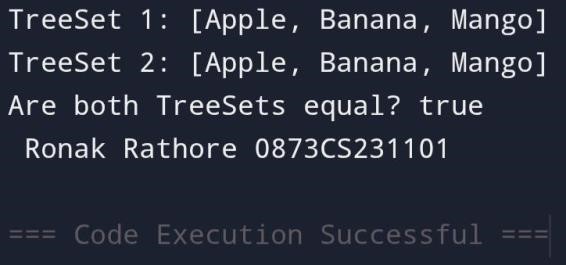
System.out.println(“TreeSet 1: “ + set1);

System.out.println(“TreeSet 2: “ + set2);

System.out.println(“Are both TreeSets equal? “ + isEqual);

System.out.println(“ Ronak Rathore 0873CS231101”);

}}



**Write a Java program to clone one HashSet into another**

Import java.util.HashSet;

Public class HashSetClone {

Public static void main(String[] args) {

HashSet<String> originalSet = new HashSet<>(); originalSet.add(“Red”); originalSet.add(“Green”); originalSet.add(“Blue”);

@SuppressWarnings(“unchecked”)

HashSet<String> clonedSet = (HashSet<String>) originalSet.clone();

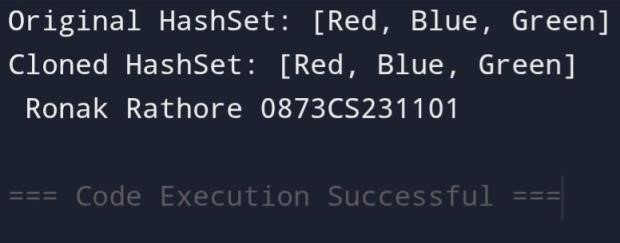
System.out.println(“Original HashSet: “ + originalSet);

System.out.println(“Cloned HashSet: “ + clonedSet);

System.out.println(“ Ronak Rathore 0873CS231101”);

}

}



**Write a Java program to convert a HashSet into an array.**

Import java.util.HashSet;

Public class HashSetToArray {

Public static void main(String[] args) {

HashSet<String> set = new HashSet<>();

Set.add(“Apple”);

Set.add(“Banana”);

Set.add(“Mango”);

String[] array = new String[set.size()];

Set.toArray(array);

System.out.println(“Elements in the array:”);

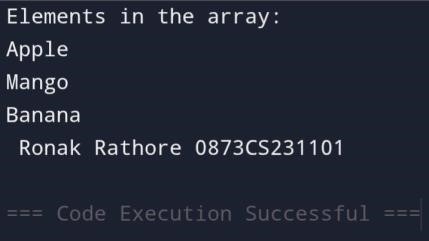
For (String element : array) {

System.out.println(element);

}

System.out.println(“ Ronak Rathore 0873CS231101”);

}}



**Write a Java program to convert a HashSet into a TreeSet**

Import java.util.HashSet;

Import java.util.TreeSet;

Public class HashSetToTreeSet {

Public static void main(String[] args) { HashSet<String> hashSet = new HashSet<>(); hashSet.add(“Banana”); hashSet.add(“Apple”); hashSet.add(“Mango”);

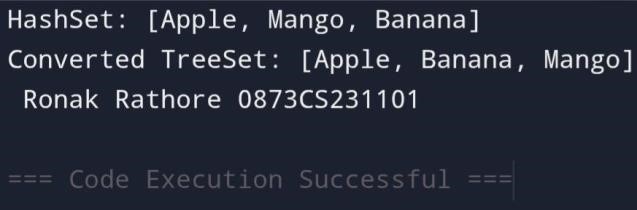
TreeSet<String> treeSet = new TreeSet<>(hashSet);

System.out.println(“HashSet: “ + hashSet);

System.out.println(“Converted TreeSet: “ + treeSet);

System.out.println(“ Ronak Rathore 0873CS231101”);

}}



**Write a Java program to find numbers less than 7 in a TreeSet**

Import java.util.TreeSet;

Public class TreeSetLessThanSeven {

Public static void main(String[] args) {

TreeSet<Integer> numbers = new TreeSet<>();

Numbers.add(1);

Numbers.add(3);

Numbers.add(5);

Numbers.add(7);

Numbers.add(9);

Numbers.add(11);

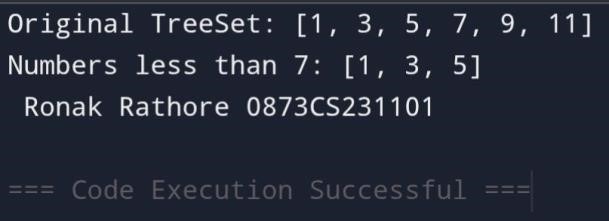
TreeSet<Integer> lessThanSeven = new TreeSet<>(numbers.headSet(7));

System.out.println(“Original TreeSet: “ + numbers);

System.out.println(“Numbers less than 7: “ + lessThanSeven);

System.out.println(“ Ronak Rathore 0873CS231101”);

}}



**Write a Java program to compare two HashSets.**

Import java.util.HashSet;

Public class HashSetCompare {

Public static void main(String[] args) {

HashSet<String> set1 = new HashSet<>();

Set1.add(“Apple”);

Set1.add(“Banana”);

Set1.add(“Mango”);

HashSet<String> set2 = new HashSet<>();

Set2.add(“Banana”);

Set2.add(“Apple”);

Set2.add(“Mango”);

Boolean isEqual = set1.equals(set2);

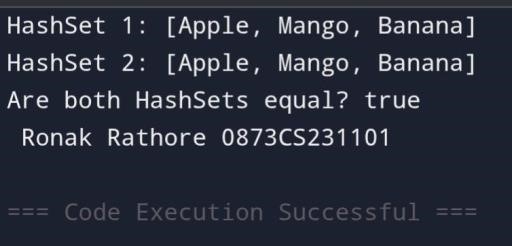
System.out.println(“HashSet 1: “ + set1);

System.out.println(“HashSet 2: “ + set2);

System.out.println(“Are both HashSets equal? “ + isEqual);

System.out.println(“ Ronak Rathore 0873CS231101”);

}}



**Write a Java program to retain common elements from two sets.**

Import java.util.HashSet;

Public class RetainCommonElements {

Public static void main(String[] args) {

HashSet<String> set1 = new HashSet<>();

Set1.add(“Apple”);

Set1.add(“Banana”);

Set1.add(“Mango”);

HashSet<String> set2 = new HashSet<>();

Set2.add(“Banana”);

Set2.add(“Mango”);

Set2.add(“Orange”);

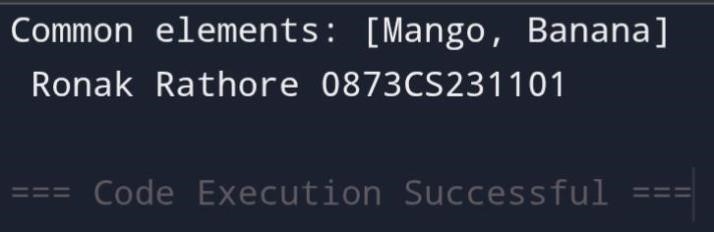
Set1.retainAll(set2);

System.out.println(“Common elements: “ + set1);

System.out.println(“Ronak Rathore 0873CS231101)”

}

}



**Write a Java program to remove all elements from a HashSet.**

Import java.util.HashSet;

Public class HashSetClear {

Public static void main(String[] args) {

HashSet<String> set = new HashSet<>();

Set.add(“Red”);

Set.add(“Green”);

Set.add(“Blue”);

System.out.println(“Original HashSet: “ + set);

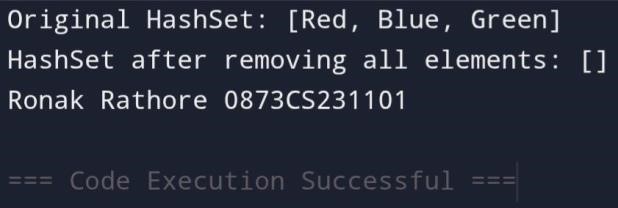
Set.clear();

System.out.println(“HashSet after removing all elements: “ + set);

System.out.println(“Ronak Rathore 0873CS231101”);

}

}



**Write a Java program to copy all mappings from one map to another.**

import java.util.HashMap; import java.util.Map; public class MapCopy {

public static void main(String[] args) {

HashMap<Integer, String> map1 = new HashMap<>(); map1.put(1, "Apple"); map1.put(2, "Banana"); map1.put(3, "Mango");

HashMap<Integer, String> map2 = new HashMap<>(); map2.putAll(map1);

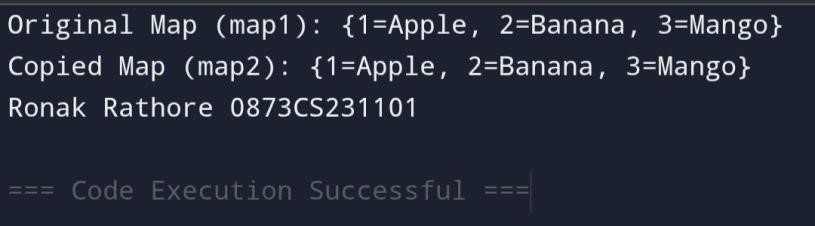
System.out.println("Original Map (map1): " + map1);

System.out.println("Copied Map (map2): " + map2);

System.out.println("Ronak Rathore 0873CS231101");

}

}



**Write a Java program to remove all key-value pairs from a map.**

Import java.util.HashMap;

Public class MapClearExample {

Public static void main(String[] args) {

HashMap<Integer, String> map = new HashMap<>();

Map.put(1, “Apple”);

Map.put(2, “Banana”);

Map.put(3, “Mango”);

System.out.println(“Original Map: “ + map);

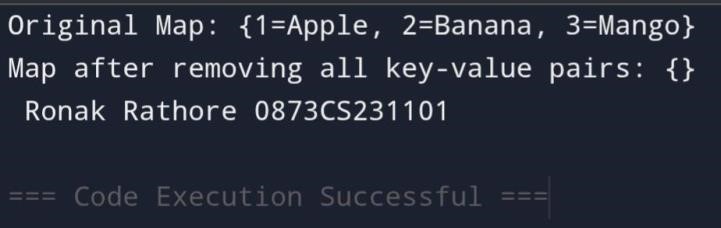
Map.clear();

System.out.println(“Map after removing all key-value pairs: “ + map);

System.out.println(“ Ronak Rathore 0873CS231101”);

}

}



**Write a Java program to check if a map is empty or contains key-value mappings.**

Import java.util.HashMap;

Public class MapIsEmptyExample {

Public static void main(String[] args) {

HashMap<Integer, String> map = new HashMap<>();

If (map.isEmpty()) {

System.out.println(“The map is empty.”);

} else {

System.out.println(“The map contains key-value mappings: “ + map);

}

Map.put(1, “Apple”);

Map.put(2, “Banana”);

If (map.isEmpty()) {

System.out.println(“The map is empty.”);

} else {

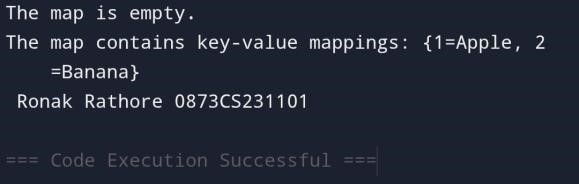
System.out.println(“The map contains key-value mappings: “ + map);

System.out.println(“ Ronak Rathore 0873CS231101”);

}

}

}



**Write a Java program to create a shallow copy of a HashMap instance.**

Import java.util.HashMap;

Public class HashMapShallowCopy {

Public static void main(String[] args) {

HashMap<Integer, String> originalMap = new HashMap<>(); originalMap.put(1, “Apple”); originalMap.put(2, “Banana”); originalMap.put(3, “Mango”);

HashMap<Integer, String> shallowCopy = (HashMap<Integer, String>) originalMap.clone();

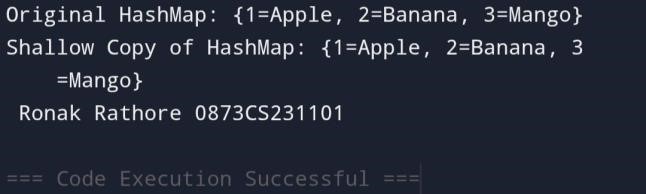
System.out.println(“Original HashMap: “ + originalMap);

System.out.println(“Shallow Copy of HashMap: “ + shallowCopy);

System.out.println(“ Ronak Rathore 0873CS231101”);

}

}



**Write a Java program to test whether a specified key exists in the map.**

Import java.util.HashMap;

Public class MapKeyCheck {

Public static void main(String[] args) {

HashMap<String, Integer> map = new HashMap<>();

Map.put(“Apple”, 10);

Map.put(“Banana”, 20);

Map.put(“Mango”, 30);

String keyToCheck = “Banana”;

If (map.containsKey(keyToCheck)) {

System.out.println(“Key ‘” + keyToCheck + “’ exists in the map.”);

} else {

System.out.println(“Key ‘” + keyToCheck + “’ does not exist in the map.”); System.out.println(“ Ronak Rathore 0873CS231101”);

}

}}

