1.Write a program for all TreeMap methods ?

```
package collection;
import java.util.*;
public class Tree {

	public static void main(String[] args) {
	TreeMap <Integer,String>employee=new
TreeMap<Integer,String>();
	employee.put(1,"Anitha");
	employee.put(2, "Anu");
	employee.put(3,"Ravina");
	employee.put(4, "Mani");
	employee.put(5, "Suresh");
	employee.put(6, "Nixen");

	System.out.println(employee);
	System.out.println(employee.clone());
	System.out.println(employee.containsKey(4));
	//System.out.println(employee.containsKey("Anu"));
	//contains ket not contains value.
	System.out.println(employee.containsValue("Ravina"));
	System.out.println(employee.containsValue(4));
	System.out.println(employee.entrySet());
	System.out.println(employee.firstEntry());
	System.out.println(employee.get(2));

	SortedMap<Integer,String> em= new
TreeMap<Integer,String>();
	em=employee.headMap(4);
	System.out.println(em);
	System.out.println("Key set:"+employee.keySet());
	System.out.println(employee.containsKey(2));
	System.out.println(employee.ceilingKey(5));
	System.out.println(employee.firstKey());
	System.out.println(employee.floorKey(5));
	System.out.println(employee.higherKey(1));
	//put key value another higher key present
```

```
        System.out.println(employee.lastKey());
        System.out.println(employee.lowerKey(2));
        System.out.println(employee.descendingKeySet());
        System.out.println(employee.navigableKeySet());

        //copy to another map

        TreeMap<Integer,String>employ = new
TreeMap<Integer,String>();
        employ.putAll(employee);
        System.out.println(employ);

        }

}
```

Output:

```
{1=Anitha, 2=Anu, 3=Ravina, 4=Mani, 5=Suresh, 6=Nixen}
{1=Anitha, 2=Anu, 3=Ravina, 4=Mani, 5=Suresh, 6=Nixen}
true
true
false
[1=Anitha, 2=Anu, 3=Ravina, 4=Mani, 5=Suresh, 6=Nixen]
1=Anitha
Anu
{1=Anitha, 2=Anu, 3=Ravina}
Key set:[1, 2, 3, 4, 5, 6]
true
5
1
5
2
6
1
[6, 5, 4, 3, 2, 1]
[1, 2, 3, 4, 5, 6]
```

{1=Anitha, 2=Anu, 3=Ravina, 4=Mani, 5=Suresh, 6=Nixen}

2.Write a program for all linkedhashmap methods?

```
package collection;

	import java.util.LinkedHashMap;
	import java.util.Map;
	import java.util.Set;

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

		LinkedHashMap<Integer, String> linkedHashMap = new
LinkedHashMap<>();

		linkedHashMap.put(1, "One");
		linkedHashMap.put(2, "Two");
		linkedHashMap.put(3, "Three");
		linkedHashMap.put(4, "Four");

		System.out.println("Original LinkedHashMap: " +
linkedHashMap);

		String value = linkedHashMap.get(2);
		System.out.println("Value associated with key 2: " +
value);

		boolean containsKey = linkedHashMap.containsKey(3);
		System.out.println("Contains key 3: " + containsKey);

		boolean containsValue =
linkedHashMap.containsValue("Five");
		System.out.println("Contains value 'Five': " +
containsValue);

		linkedHashMap.remove(4);
```

```
            System.out.println("LinkedHashMap after removing key 4:
" + linkedHashMap);

            Set<Integer> keys = linkedHashMap.keySet();
            System.out.println("Keys in the LinkedHashMap: " +
keys);

            for (String val : linkedHashMap.values()) {
                System.out.println("Value: " + val);
            }

            System.out.println("Iterating through the LinkedHashMap
using a for-each loop:");
            for (Map.Entry<Integer, String> entry :
linkedHashMap.entrySet()) {
                int key = entry.getKey();
                String val = entry.getValue();
                System.out.println("Key: " + key + ", Value: " + val);
            }

            linkedHashMap.clear();
            System.out.println("LinkedHashMap after clearing: " +
linkedHashMap);
        }
    }
```

Output:

```
Original LinkedHashMap: {1=One, 2=Two, 3=Three, 4=Four}
Value associated with key 2: Two
Contains key 3: true
Contains value 'Five': false
LinkedHashMap after removing key 4: {1=One, 2=Two, 3=Three}
Keys in the LinkedHashMap: [1, 2, 3]
Value: One
Value: Two
Value: Three
Iterating through the LinkedHashMap using a for-each loop:
Key: 1, Value: One
```

```
Key: 2, Value: Two
Key: 3, Value: Three
LinkedHashMap after clearing: {}
```

3. Extract Digits/ Numbers from String using Java Program

```
package collection;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import java.util.Scanner;
        public class Extract {

            public static void main(String[] args) {

                Scanner scanner = new Scanner(System.in);

                System.out.print("Enter a string that contains numbers: ");
                String input = scanner.nextLine();

                scanner.close();

                Pattern pattern = Pattern.compile("\\d+");

                Matcher matcher = pattern.matcher(input);

                StringBuilder numbers = new StringBuilder();

                while (matcher.find()) {
                    numbers.append(matcher.group());
                }

                if (numbers.length() > 0) {
                    System.out.println("Numbers are: " +
numbers.toString());
                } else {
                    System.out.println("No numbers found in the input
string.");
```

```
        }
    }
}
```

Output:

Enter a string that contains numbers: 38729dsah;kjshs;jk33232
Numbers are: 3872933232

4. Given a sentence (string) and we have to extract words from a string using java program.

```
package collection;
import java.util.Scanner;
public class Words {
  public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);


      System.out.print("Enter a string that contains letters and
numbers: ");
                String input = scanner.nextLine();

                scanner.close();
                String lettersOnly = input.replaceAll("[^a-zA-Z]",
"");

                System.out.println("Letters extracted from the string:
" + lettersOnly);
              }
            }
```

Ouput:

Enter a string that contains letters and numbers:
7429hjdskjhakj323jh2
Letters extracted from the string: hjdskjhakjjh

5. Remove duplicate elements from unsorted array.

```
package collection;
import java.util.HashSet;
import java.util.ArrayList;
import java.util.Arrays;

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

            int[] arr = {5, 2, 8, 7, 2, 1, 2, 2, 5, 9, 8, 3};

            HashSet<Integer> uniqueElements = new HashSet<>();
            ArrayList<Integer> result = new ArrayList<>();

            for (int element : arr) {
               if (uniqueElements.add(element)) {
                  result.add(element);
               }
            }

            // Convert the ArrayList back to an array (if needed)
            int[] resultArray = new int[result.size()];
            for (int i = 0; i < result.size(); i++) {
               resultArray[i] = result.get(i);
            }

            // Display the array with duplicates removed
            System.out.println("Array with duplicates removed: " +
Arrays.toString(resultArray));
        }
    }
```

Ouput:

```
Array with duplicates removed: [5, 2, 8, 7, 1, 9, 3]
```

6. Find common elements in three sorted arrays.

```
package collection;

public class Three {
	public static void main(String[] args) {
	   int[] arr1 = {1, 3, 4, 5, 7};
	   int[] arr2 = {2, 3, 5, 6};
	   int[] arr3 = {3, 5, 8};

	   find(arr1, arr2, arr3);
	}

	public static void find(int[] arr1, int[] arr2, int[] arr3) {
	   int i = 0, j = 0, k = 0;

	   while (i < arr1.length && j < arr2.length && k < 
arr3.length) {
	      if (arr1[i] == arr2[j] && arr2[j] == arr3[k]) {
	         System.out.println("Common element: " + arr1[i]);
	         i++;
	         j++;
	         k++;
	      } else if (arr1[i] < arr2[j]) {
	         i++;
	      } else if (arr2[j] < arr3[k]) {
	         j++;
	      } else {
	         k++;
	      }
	   }
	}
}
```

Ouput:

Common element: 3
Common element: 5