Assingment 13.

Part A:

1. Difference between HashSet and TreeSet

HashSet and TreeSet both implement the Set interface and store unique elements.

HashSet:
- Uses a hash table internally.
- Does not maintain insertion order.
- Allows one null element.
- Provides O(1) performance for basic operations.

TreeSet:
- Uses a Red-Black Tree internally.
- Maintains elements in sorted order.
- Does not allow null elements.
- Provides O(log n) performance.

HashSet is preferred when ordering is not required, while TreeSet is used when sorted data is needed.

2. What is Map Interface?

The Map interface stores elements in key-value pairs where keys are unique and values can be duplicated. It is part of java.util and does not extend the Collection interface.

Common implementations include HashMap, Hashtable, LinkedHashMap, and TreeMap.

Maps are useful for fast data retrieval using keys.

3. Difference between HashMap and Hashtable

HashMap:
- Not synchronized
- Faster
- Allows one null key and multiple null values

Hashtable:
- Synchronized (thread-safe)
- Slower
- Does not allow null key or value

HashMap is commonly used in modern Java applications.

4. What is LinkedHashMap?

LinkedHashMap is a subclass of HashMap that maintains insertion order.

It uses both a hash table and a doubly linked list to store elements.

It is useful when predictable iteration order is required.

5. What is TreeMap?

TreeMap stores key-value pairs in sorted order of keys.

It uses a Red-Black Tree internally and implements the SortedMap interface.

TreeMap does not allow null keys and provides O(log n) performance.

Part B:

1. Program to compare two ArrayLists

```
import java.util.ArrayList;

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

      ArrayList<Integer> list1 = new ArrayList<>();
      list1.add(10);
      list1.add(20);
      list1.add(30);

      ArrayList<Integer> list2 = new ArrayList<>();
      list2.add(10);
      list2.add(20);
      list2.add(30);

      if (list1.equals(list2)) {
         System.out.println("Both ArrayLists are equal");
      } else {
         System.out.println("ArrayLists are not equal");
      }
   }
}
```

2. Program to find frequency of elements in ArrayList

```
import java.util.ArrayList;
import java.util.HashMap;

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

      ArrayList<Integer> list = new ArrayList<>();
      list.add(10);
      list.add(20);
      list.add(10);
      list.add(30);
      list.add(20);
      list.add(10);

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

      for (Integer num : list) {
         map.put(num, map.getOrDefault(num, 0) + 1);
      }

      System.out.println(map);
   }
}
```

3. Program to convert ArrayList to array

```
import java.util.ArrayList;

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

      ArrayList<String> list = new ArrayList<>();
      list.add("Java");
      list.add("Python");
      list.add("C++");

      String[] arr = list.toArray(new String[0]);

      for (String s : arr) {
         System.out.println(s);
      }
   }
}
```

4. Program to convert array to ArrayList

```
import java.util.ArrayList;
import java.util.Arrays;

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

      String[] arr = {"A", "B", "C"};

      ArrayList<String> list = new ArrayList<>(Arrays.asList(arr));

      System.out.println(list);
   }
}
```

5. Program to use HashSet to remove duplicates from a list

```
import java.util.ArrayList;
import java.util.HashSet;

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

      ArrayList<Integer> list = new ArrayList<>();
      list.add(10);
      list.add(20);
      list.add(10);
      list.add(30);
      list.add(20);

      HashSet<Integer> set = new HashSet<>(list);

      System.out.println(set);
   }
}
```