Sabbir Ahamad
IT22007

1. Write a program to find the k^{th} smallest element in an Arraylist.

Soln:

```
import java.util.*;
public class KthSmallest{
  public static int findKthSmallest(List<Integer>list,
                                    int k){
    Collections.sort(list);
    return list.get(k-1);
  }
  public static void main(String[] args){
    List<Integer> numbers = Arrays.asList(7,10,4,3,20);
    int k=3;
    int result = findKthSmallest(numbers,k);
    System.out.println("The " + k + "rd smallest elemnt
                       is: " + result);

  }
}
```

2. Create a TreeMap to store the mappings of words to their frequencies in a given text.

Soln:

```
import java.util.*;
public class WordFrequencyMap{
 public static void main(String [] args){
 String text = "hello world. Hello Java";
 String[] words = text.split(" ");
 TreeMap<String, Integer> frequencyMap = new
                                    TreeMap<>();

  for (String word: Words)
  {
   frequencyMap.put(word, frequencyMap.getOrDefault
                                    (word,0)+1);

  }
  System.out.println("Word Frequencies: ");
 for(Map.Entry<String, Integer> entry: frequency-
            frequencyMap.entrySet()){
   System.out.println(entry.getKey()"": " +
                                    entry.getValue());

  }
}
```

3] Queue and Stack using Priority Queue with a custom comparator.

```
import java.util.*;
public class QueueStackPQ{
  public static void main(String[] args){
    PriorityQueue<Integer> queue = new PriorityQueue
                                        <>();
      queue.add(3);
      queue.add(1);
      queue.add(2);
    System.out.println("Queue(ascending order): ");
    while(!queue.isEmpty()){
      System.out.print(queue.poll()+" ");
    }
  PriorityQueue<Integer> stack = new PriorityQueue
                                  <>(Collection.reverseOrder());
    stack.add(1);
    stack.add(2);
    stack.add(3);
  System.out.println("\n Stack(descending order): ");
```

```
        while(!stack.empty()){
         System.out.print(stack.poll() + " ");
        }
    }
}
```

4] TreeMap to store students IDs and their details.

Soln:

```
import java.util.*;
class Student{
    String name;
    int age;
    Student(String name, int age){
        this.name = name;
        this.age = age;
    }
    public String toString(){
        return name + "(" "target.");
    }
}
```

```
Public class StudentMap{
  public static void main(String[] args){
    TreeMap<Integer, Student> Students = new TreeMap<>();
    Students.put(101, new Student("Adam", 20));
    Students.put(102, new Student("Anib", 21));
    for(Map.Entry<Integer, Student> entry : students.
                                        entrySet()){
      System.out.println("ID: " + entry.getKey() + ",
                         Details: " + entry.getValue());
    }
  }
}
```

5] Check if two linkedlists are equal

Soln:

```
import java.util.*;
public class LinkedListEqual{
  public static void main(String[] args){
    LinkedList<Integer> list1 = new LinkedList<>(Arrays.
                                        asList(1,2,3));
    LinkedList<Integer> list2 = new LinkedList<>(Arrays.
                                        asList(1,2,3));
```

```
        boolean isEqual = list1.equals(list2);
        System.out.println("Are the lists equal? " +
                                        isEqual);
    }
}
```

6) HashMap for employee ID to department.

Soln:

```
import java.util.*;
public class EmployeeDept{
  public static void main(String[] args){
    HashMap<Integer, String> empDept = new HashMap<>();
    empDept.put(1001, "HR");
    empDept.put(1006, "CSE");
    empDept.put(1005, "Accounting");
    for (Map.Entry<Integer, String> entry: empDept.
                                        entrySet()){
      System.out.println("Employee ID: " + entry() +
        ", Department: " + entry.getValue());
}}}
```