1.All Vowels

Write a Program to check if given word contains exactly five vowels and the vowels are in alphabetical order. Return 1 if the condition is satisfied else return -1. Assume there is no repetition of any vowel in the given string and all letters are in lower case.
Include a class UserMainCode with a static method testOrderVowels which accepts a string The return type is integer based on the condition stated above.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string with maximum size of 100 characters.
Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

acebisouzz

Sample Output 1:

valid

Sample Input 2:

alphabet

Sample Output 2:

invalid

Note: Need to implement this by using LinkedHashSet
Source code:

```
package Assignment14;

import java.util.LinkedHashMap;
import java.util.Scanner;
public class Vowels {
public static void main(String[] args) {

 Scanner scan=new Scanner(System.in);
```

```
 UserMainCode umc=new UserMainCode();
 String a=scan.next();
 int ans1=umc.vowels(a);
 if(ans1==1) {
 System.out.println("Valid");
 }
 else {
 System.out.println("Invalid");
 }
 String b=scan.next();
 int ans2=umc.vowels(b);
 if(ans2==1) {
 System.out.println("Valid");
 }
 else {
 System.out.println("Invalid");
 }
}
}
class UserMainCode{
public int vowels(String s) {
 int isValid = 0;
 LinkedHashMap<Character, Integer> map = new LinkedHashMap<>();
 for(int i = 0; i < s.length(); i++) {
 char c = s.charAt(i);
 if(c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {
 if(!map.containsKey(c)) {
 map.put(c, 1);
 }
 }
 }
 if(map.size() == 5) {
 isValid = 1;
 }
 return isValid;
}
}
```

2.Employee Bonus

A Company wants to give away bonus to its employees. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

1.
Read Employee details from the User. The details would include id, DOB (date of birth) and salary in the given order. The datatype for id is integer, DOB is string and salary is integer.

2.
You decide to build two hashmaps. The first hashmap contains employee id as key and DOB as value, and the second hashmap contains same employee ids as key and salary as value.

3.
If the age of the employee in the range of 25 to 30 years (inclusive), the employee should get bonus of 20% of his salary and in the range of 31 to 60 years (inclusive) should get 30% of his salary. store the result in TreeMap in which Employee ID as key and revised salary as value. Assume the age is caculated based on the date 01-09-2014. (Typecast the bonus to

integer).
4.
Other Rules:
a. If Salary is less than 5000 store -100.
b. If the age is less than 25 or greater than 60 store -200.
c. a takes more priority than b i.e both if a and b are true then store -100.
5.
You decide to write a function calculateRevisedSalary which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.

Input and Output Format:

Input consists of employee details. The first number indicates the size of the employees. The next three values indicate the employee id, employee DOB and employee salary. The Employee DOB format is "dd-mm-yyyy"

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

2

1010

20-12-1987

10000

2020

01-01-1985

14400

Sample Output 1:

1010

12000

2020

17280

Note: Need to use the HashMap and TreeMap

```
package Assignment14;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
import java.util.TreeMap;
public class EmployeeBonus {
 public static void main(String[] args) throws ParseException{
 Scanner sc = new Scanner(System.in);
 int size = sc.nextInt();
 HashMap<Integer, String> map1 = new HashMap<>();
 HashMap<Integer, Integer> map2 = new HashMap<>();
 for(int i=0;i<size;i++){
 int id = sc.nextInt();
 String date = sc.next();
 int salary = sc.nextInt();
 map1.put(id, date);
 map2.put(id, salary);
 }
 UserMainCode1.calculateRevisedSalary(map1, map2);
 }
}
class UserMainCode1{
 static void calculateRevisedSalary(HashMap<Integer, String> map1,
HashMap<Integer, Integer>
map2){
 DateFormat df = new SimpleDateFormat("dd-MM-yyyy");
 Calendar c = Calendar.getInstance();
 TreeMap<Integer, Integer> revisedSalaryMap = new TreeMap<>();
 try{
 c.setTime(df.parse("01-09-2014"));
 }catch(ParseException e){
 System.out.println("Invalid date format");
 }
 for(Map.Entry<Integer, String> entry : map1.entrySet()){
 int key = entry.getKey();
 String date = entry.getValue();
 Calendar birthdate = Calendar.getInstance();
 try{
 birthdate.setTime(df.parse(date));
 }catch(ParseException e){
 System.out.println("Invalid date format");
 }
 int age = c.get(Calendar.YEAR) - birthdate.get(Calendar.YEAR);
 int salary = map2.get(key);
 if(salary < 5000){
 revisedSalaryMap.put(key, -100);
 }
 else if(age < 25 || age > 60){
 revisedSalaryMap.put(key, -200);
 }
 else if(age>=25 && age<=30){
 revisedSalaryMap.put(key, (int)(salary+(salary*0.2)));
```

```
 }
 else{
 revisedSalaryMap.put(key, (int)(salary+(salary*0.3)));
 }
 }
 for(Map.Entry<Integer, Integer> entry : revisedSalaryMap.entrySet()){
 System.out.println(entry.getKey()+" "+entry.getValue());
 }
 }
}
```

3.Largest Key in HashMap

Write a program that construts a hashmap and returns the value corresponding to the largest key.

Include a class UserMainCode with a static method getMaxKeyValue which accepts a string. The return type (String) should be the value corresponding to the largest key.

Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode.

Input and Output Format:

Input consists of 2n+1 values. The first value corresponds to size of the hashmap. The next n pair of numbers equals the integer key and value as string.

Output consists of a string which is the value of largest key.

Refer sample output for formatting specifications.

Sample Input 1:

3

12

amron

9

Exide

7

SF

Sample Output 1:

Amron

Source code:

```
package Assignment14;
```

```
import java.util.Collections;
import java.util.Map;
public class Largestkey {
public static String getMaxkeyValue(Map<Integer, String> hashmap) {
String maxValue = (Collections.max(hashmap.values()));
for (Map.Entry<Integer, String> key : hashmap.entrySet()) {
if (key.getValue() == maxValue) {
return key.getKey() + ":" + key.getValue();
}
}
return null;
}
}
```

```
package Assignment14;

import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

public class TestLargestKey {
public static void main(String[] args) {
// TODO Auto-generated method stub
Largestkey largestkey = new Largestkey();
Map<Integer, String> hashmap = new HashMap<>();
System.out.println("Enter the size ");
Scanner sc = new Scanner(System.in);
int size = sc.nextInt();
System.out.println("Enter key and the value");
for (int i = 1; i <= size; i++) {
hashmap.put(sc.nextInt(), sc.next());
}
System.out.println(Largestkey.getMaxkeyValue(hashmap));
}
}
```