1. Unique Even Sum

Write a program to read an array, eliminiate duplicate elements and calculate the sum of even numbers (values) present in the array.

Include a class UserMainCode with a static method **addUniqueEven** which accepts a single integer array. The return type (integer) should be the sum of the even numbers. In case there is no even number it should return -1.

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

Input and Output Format:

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

In case there is no even integer in the input array, print **no even numbers** as output. Else print the sum.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

Sample Input 1:

4

2

5

1

4

Sample Output 1:

6

Sample Input 2:

3

1

1

1

Sample Output 2:

no even numbers

Solutions:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
int n = sc.nextInt();
int[] a = newint[20];
for (int i = 0; i < n; i++)
a[i] = sc.nextInt();
int res = User.addUniqueEven(a);
if (res == -1)
System.out.println("no even numbers");
else
System.out.println(res);

}
}


publicclass User {
publicstaticint addUniqueEven(int a[]) {
int i = 0, j = 0, count = 0, sum = 0;
int n = a.length;
for (i = 0; i < n; i++) {
count = 0;
for (j = i + 1; j < n; j++) {
if (a[i] == a[j])
count++;
}
if (count == 0) {
if (a[i] % 2 == 0)
sum = sum + a[i];
}
}
if (sum == 0)
```

```
return -1;
else
returnsum;
}
}
```

2. Palindrome & Vowels

Write a program to check if a given string is palindrome and contains at least two different vowels.

Include a class UserMainCode with a static method **checkPalindrome** which accepts a string. The return type (integer) should be 1 if the above condition is satisfied, otherwise return -1.

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

Note – Case Insensitive while considering vowel, i.e a &A are same vowel, But Case sensitive while considering palindrome i.e abc CbA are not palindromes.

Input and Output Format:

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

Output consists of a single Integer.

Refer sample output for formatting specifications.

Sample Input 1:

abceecba

Sample Output 1:

valid

Sample Input 2:

abcd

Sample Output 2:

Invalid

Solution :

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s = sc.next();
int res = User.checkPalindrome(s);
if (res == 1)
System.out.println("valid");
else
System.out.println("invalid");

}
}


publicclass User {
publicstaticint checkPalindrome(String s) {
int res = 0, i = 0, j = 0, count = 0, k = 0;
StringBuffer sb = new StringBuffer(s);
sb.reverse();
if (sb.toString().equals(s)) {
for (i = 0; i < s.length(); i++) {
count = 0;

for (j = i + 1; j < s.length(); j++) {
if (s.charAt(i) == s.charAt(j))
count++;
}
if (count == 0)
if (s.charAt(i) == 'a' || s.charAt(i) == 'e'
|| s.charAt(i) == 'i' || s.charAt(i) == 'o'
|| s.charAt(i) == 'u' || s.charAt(i) == 'A'
|| s.charAt(i) == 'E' || s.charAt(i) == 'I'
|| s.charAt(i) == 'O' || s.charAt(i) == 'U')
k++;
}
}
if (k >= 2)
res = 1;
else
res = 0;

return res;
}
```

}

3. Strings – Unique & Existing Characters

Obtain two strings from user as input. Your program should modify the first string such that all the characters are replaced by plus sign (+) except the characters which are present in the second string.

That is, if one or more characters of first string appear in second string, they will not be replaced by +.

Return the modified string as output. Note - ignore case.

Include a class UserMainCode with a static method **replacePlus** which accepts two string variables. The return type is the modified string.

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 two strings with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

abcxyz

axdef

Sample Output 1:

a++ x++

Sample Input 2:

ABCDEF

feCBAd

Sample Output 2:

ABCDEF

Solution:

```
import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = newScanner(System.in);
String s1 = sc.nextLine();
String s2 = sc.nextLine();
System.out.println(User.replacePlus(s1, s2));
}
}

publicclass User {
publicstatic String replacePlus(String s1, String s2) {
String ss1 = s1.toLowerCase();
String ss2 = s2.toLowerCase();
StringBuffer sb = newStringBuffer();
for (int i = 0; i < s1.length(); i++) {
char c = ss1.charAt(i);
if (ss2.indexOf(c) == -1)
sb.append('+');
else
sb.append(s1.charAt(i));
}
return sb.toString();
}
}
```

4. Longest Word

Write a Program which finds the longest word from a sentence. Your program should read a sentence as input from user and return the longest word. In case there are two words of maximum length return the word which comes first in the sentence.

Include a class UserMainCode with a static method **getLargestWord** which accepts a string The return type is the longest word of type string.

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:

Welcome to the world of Programming

Sample Output 1:

Programming

Sample Input 2:

ABC DEF

Sample Output 2:

ABC

Solution:

```
import java.util.Scanner;

publicclass Main {

	publicstaticvoid main(String[] args) {
		Scanner s = newScanner(System.in);
		String s1 = s.nextLine();
		System.out.println(User.getLongestWord(s1));
	}

}

publicclass User {
	publicstatic String getLongestWord(String s) {
```

```
            int len, i, p = 0, max = 0, count = 0;
            char b;
            s = s.concat(" ");
            len = s.length();
            for (i = 0; i < len; i++) {
                  b = s.charAt(i);
                  if (b != ' ') {
                        count++;
                  } else {
                        if (count > max) {
                              max = count;
                              p = i;
                        }
                        count = 0;
                  }
            }
            return (s.substring(p - max, p));
      }
}
```

```
import java.util.Scanner;
public class PalindromeMain {

      public static void main(String[] args) {

            Scanner sc = new Scanner(System.in);
            String s1 = sc.nextLine();

            System.out.println(Palindrome.checkPalindrome(s1));
                  }

}
```

```
import java.util.StringTokenizer;


public class Palindrome {

      public static String checkPalindrome(String s1)
      {
    int res,max=0;
    String s2=null;
      StringTokenizer st=new StringTokenizer(s1," ");
      while(st.hasMoreTokens())
      {
            String s=st.nextToken();
            res=s.length();
            if(res>max)
            {
```

```
                max=res;

        s2=s;

        }
}
        return s2;

}
        }
```

5. String Occurences

Obtain two strings from user as input. Your program should count the number of occurences of second word of second sentence in the first sentence.

Return the count as output. Note - Consider case.

Include a class UserMainCode with a static method **countNoOfWords** which accepts two string variables. The return type is the modified string.

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 two strings with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

abc bcd abc bcd abc abc

av abc

Sample Output 1:

4

Sample Input 2:

ABC xyz AAA

w abc

Sample Output 2:

0

Solution:

```
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
String s1 = s.nextLine();
String s2 = s.nextLine();
System.out.println(User.countNoOfWords(s1, s2));
}
}
```

```
import java.util.StringTokenizer;
publicclass User {
publicstaticint countNoOfWords(String s1, String s2) {
String[] a = new String[s1.length()];
String[] b = new String[s2.length()];
int i = 0, j = 0, count = 0;
StringTokenizer st1 = newStringTokenizer(s1, " ");
StringTokenizer st2 = newStringTokenizer(s2, " ");

while (st1.hasMoreTokens()) {
a[i] = st1.nextToken();
i++;
}
while (st2.hasMoreTokens()) {
b[j] = st2.nextToken();
j++;
}
for (int k = 0; k < i; k++) {
if (b[1].equals(a[k])) {
count++;
}
}
return count;
}
}
```

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

            Scanner sc = new Scanner(System.in);
            String s1 = sc.nextLine();
            String s2 = sc.nextLine();
            System.out.println(Palindrome.checkPalindrome(s1,s2));
                  }

}
```

```
import java.util.StringTokenizer;


public class Palindrome {

      public static int checkPalindrome(String s1,String s2)
      {

    int count=0;
      StringTokenizer st=new StringTokenizer(s1," ");
      StringTokenizer st1=new StringTokenizer(s2," ");
      String a2=st1.nextToken();
      String b2=st1.nextToken();

      while(st.hasMoreTokens())
      {
            String s=st.nextToken();

            if(s.equalsIgnoreCase(b2))
            {
                  count++;

            }
      }
            return count;

      }
            }
```

6. ArrayList Manipulation

Write a program that performs the following actions:

1. Read 2n integers as input.
2. Create two arraylists to store n elements in each arraylist.
3. Write a function **generateOddEvenList** which accepts these two arraylist as input.
4. The function fetch the odd index elements from first array list and even index elements from second array list and add them to a new array list according to their index.
5. Return the arraylist.

Include a class UserMainCode with the static method **generateOddEvenList** which accepts two arraylist and returns an arraylist.

Create a Class Main which would be used to read 2n integers and call the static method present in UserMainCode.

Note:

- The index of first element is 0.

- Consider 0 as an even number.

- Maintain order in the output array list

Input and Output Format:

Input consists of 2n+1 integers. The first integer denotes the size of the arraylist, the next n integers are values to the first arraylist, and the last n integers are values to the second arraylist.

Output consists of a modified arraylist as per step 4.

Refer sample output for formatting specifications.

Sample Input 1:

5

12

13

14

15

16

2

3

4

5

6

Sample Output 1:

2

13

4

15

6

Solution :

```
import java.util.ArrayList;
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
int n = s.nextInt();
ArrayList<Integer> al1 = new ArrayList<Integer>();
ArrayList<Integer> al2 = new ArrayList<Integer>();
ArrayList<Integer> a = new ArrayList<Integer>();
for (int i = 0; i < n; i++)
al1.add(s.nextInt());
for (int i = 0; i < n; i++)
al2.add(s.nextInt());
a = User.generateOddEvenList(al1, al2);
for (inti = 0; i< a.size(); i++)
System.out.println(a.get(i));
}
}
```

```
import java.util.ArrayList;

publicclass User {
publicstatic ArrayList<Integer> generateOddEvenList(ArrayList<Integer> a1,
ArrayList<Integer> a2)
```

```
{
ArrayList<Integer> a = new ArrayList<Integer>();
int i = 0;
for (i = 0; i < a1.size(); i++) {
if (i % 2 == 0)
a.add(a2.get(i));
else
a.add(a1.get(i));
}
return a;

}
}
```

7. Duplicate Characters

Write a Program which removes duplicate characters from the string. Your program should read a sentence (string) as input from user and return a string removing duplicate characters. Retain the first occurance of the duplicate character. Assume the characters are case – sensitive.

Include a class UserMainCode with a static method **removeDuplicates** which accepts a string. The return type is the modified sentence of type string.

Create a Class Main which would be used to accept the input string 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:

hi this is sample test

Sample Output 1:

hi tsample

Sample Input 2:

ABC DEF

Sample Output 2:

ABC DEF

Solution:

```
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
String ss = s.nextLine();
System.out.println(User.removeDuplicates(ss));
}
}


import java.util.Iterator;
importjava.util.LinkedHashSet;

publicclass User {
publicstatic String removeDuplicates(String s) {
char a[] = s.toCharArray();
StringBuffer sb = newStringBuffer();
LinkedHashSet<Character> lh = newLinkedHashSet<Character>();
for (int i = 0; i < a.length; i++)
lh.add(a[i]);
Iterator<Character> itr = lh.iterator();
while (itr.hasNext()) {
char c = itr.next();
if (c != ' ')
;
sb.append(c);
}
return sb.toString();
}
}
```

```
import java.util.Scanner;
class Main
{	public static void main(String[] arg)
	{
		Scanner sc=new Scanner(System.in);
		String s=sc.nextLine();
	System.out.println(MainClass.removeDuplicate(s));
	}}



import java.util.HashSet;
import java.util.LinkedHashSet;
public class MainClass {

	public static  String removeDuplicate(String s)
	{
	  LinkedHashSet<Character> has=new LinkedHashSet<Character>();
		for(int i=0;i<s.length();i++)
		{
			has.add(s.charAt(i));
		}
		StringBuffer sb=new StringBuffer();
		for(Character c:has)
		{
			sb.append(c);
		}
		return sb.toString();
}
}
```

8. Mastering Hashmap

You have recently learnt about hashmaps and in order to master it, you try and use it in all of your programs.

Your trainer / teacher has given you the following exercise:

1. Read 2n numbers as input where the first number represents a key and second one as value. Both the numbers are of type integers.
2. Write a function **getAverageOfOdd** to find out average of all values whose keys are represented by odd numbers. Assume the average is an int and never a decimal number. Return the average as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read 2n numbers and build the hashmap. Call the static method present in UserMainCode.

Input and Output Format:

Input consists of a 2n+ 1 integers. The first integer specifies the value of n (essentially the hashmap size). The next pair of n numbers denote the key and value.

Output consists of an integer representing the average.

Refer sample output for formatting specifications.

Sample Input 1:

4

2

34

1

4

5

12

4

22

Sample Output 1:

8

Solution:

```
import java.util.HashMap;
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
int n = s.nextInt();
HashMap<Integer, Integer> hm1 = new HashMap<Integer, Integer>();
for (int i = 0; i < n; i++)
hm1.put(s.nextInt(), s.nextInt());
System.out.println(User.getAverageOfOdd(hm1));
}
}
```

```
import java.util.HashMap;
import java.util.Iterator;

publicclass User {
publicstaticint getAverageOfOdd(HashMap<Integer, Integer> hm1) {
int sum = 0, count = 0;
Iterator<Integer> itr = hm1.keySet().iterator();
while (itr.hasNext()) {
int key = itr.next();
if (key % 2 != 0) {
count++;
int val = hm1.get(key);
sum = sum + val;
}
}
int avg = sum / count;
return avg;
}
}
```

9. **Managers & Hashmaps**

A Company wants to automate its payroll process. You have been assigned as the programmer to build this package. 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, designation and salary in the given order. The datatype for id is integer, designation is string and salary is integer.
2. You decide to build two hashmaps. The first hashmap contains employee id as key and designation as value, and the second hashmap contains same employee ids as key and salary as value.
3. The company decides to hike the salary of managers by 5000. You decide to write a function **increaseSalaries** which takes the above hashmaps as input and returns a hashmap with only managers id and their increased salary 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 designation and employee salary.

Output consists of a single string.

Refer sample output for formatting specifications.

Sample Input 1:

2

2

programmer

3000

8

manager

50000

Sample Output 1:

8

55000

Solution :

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.Scanner;

class Main {
public static void main(String args[]) {
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
HashMap<Integer, String> h1 = new HashMap<Integer, String>();
HashMap<Integer, Integer> h2 = new HashMap<Integer, Integer>();
HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();
for (int i = 0; i < n; i++) {
int id = sc.nextInt();
h1.put(id, sc.next());
h2.put(id, sc.nextInt());
}
hm = User.dis(n, h1, h2);
Iterator<Integer> itr = hm.keySet().iterator();
while (itr.hasNext()) {
int id = itr.next();
int sal = hm.get(id);
System.out.println(id);
System.out.println(sal);
}
}
}
```

```
import java.util.HashMap;
import java.util.Iterator;

public class User {
public static HashMap<Integer, Integer> dis(int n,
HashMap<Integer, String> h1, HashMap<Integer, Integer> h2) {
HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();

Iterator<Integer> itr = h1.keySet().iterator();
while (itr.hasNext()) {
int id = itr.next();
String deg = h1.get(id);
```

```
if (deg.equalsIgnoreCase("manager")) {
hm.put(id, h2.get(id) + 5000);
}
}
return hm;

}
}
```

```
  import java.util.HashMap;
import java.util.Iterator;
import java.util.Scanner;
public class Main {
public static void main(String[] args)
{

      Scanner sc=new Scanner(System.in);

      HashMap<Integer,String> ip1=new  HashMap<Integer,String>();
      HashMap<Integer,Integer> ip2=new  HashMap<Integer,Integer>();
      int n=Integer.parseInt(sc.nextLine());

      for(int i=0;i<n;i++)
      {
            int id=Integer.parseInt(sc.nextLine());


      ip1.put(id,sc.nextLine());

      ip2.put(id,Integer.parseInt(sc.nextLine()));
      }

      HashMap<Integer,Integer> op=new  HashMap<Integer,Integer>();
      op=MainClass.addsal(ip1,ip2);

      Iterator<Integer> itr=op.keySet().iterator();
      while(itr.hasNext())
      {
            int  key=itr.next();
```

```
            int value=op.get(key);
            System.out.println(key);
            System.out.println(value);

      }


}}

/*

int n=sc.nextInt();
      for(int i=0;i<n;i++)
      {
      int id=sc.nextInt();
      ip1.put(id,sc.next());
      ip2.put(id,sc.nextInt());
      }

*/


import java.util.HashMap;
import java.util.Iterator;

public class MainClass {
      public static HashMap<Integer,Integer>
addsal(HashMap<Integer,String> hm1,
                  HashMap<Integer,Integer> hm2)
                  {
             HashMap<Integer,Integer>op=new
HashMap<Integer,Integer>();

             Iterator<Integer> itr=hm1.keySet().iterator();

             while(itr.hasNext())
             {
                   int id=itr.next();
                   String s=hm1.get(id);
                   if(s.equals("manager"))
                   {
                         int newsal=hm2.get(id)+5000;
                         op.put(id,newsal);
                   }

             }
             return op;

                        }
}
```

10. Check first and last word

Write a program to check if the first word and the last word in the input string match.

Include a class **UserMainCode** with a static method **“check”** that accepts a string argument and returns an int. If the first word and the last word in the string match, the method returns the number of characters in the word. Else the method returns the sum of the number of characters in the first word and last word.

Create a class **Main** which would get the input as a String and call the static method **check** present in the UserMainCode.

Input and Output Format:
Input consists of a string.
Output is an integer.

Sample Input 1:
how are you you are how

Sample Output 1:
3

Sample Input 2:
how is your child

Sample Output 2:
8

Solution:

```
import java.util.Scanner;

publicclass Main {
```

```
publicstaticvoid main(String[] args) {
Scanner s = newScanner(System.in);
String ss = s.nextLine();
System.out.println(User.check(ss));
}
}
```

```
import java.util.StringTokenizer;

publicclass User {
publicstaticint check(String s) {
StringTokenizer st = new StringTokenizer(s, " ");
int n = st.countTokens();
String[] s1 = new String[n];
int i = 0, value = 0;
while (st.hasMoreTokens()) {
s1[i] = st.nextToken();
i++;
}
if (s1[0].equals(s1[i - 1]))
value = s1[0].length();
else
value = s1[0].length() + s1[i - 1].length();
return value;
}
}
```