Set-3

1.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

UserMainCode

```
import java.util.HashMap;
import java.util.Iterator;
public class UserMainCode {
public static int getAverageOfOdd(HashMap<Integer,Integer>h1)
{
int av=0,c=0,s=0;
Iterator<Integer> it=h1.keySet().iterator();
while(it.hasNext())
{
int a=it.next();
if(a%2!=0)
{
int b=h1.get(a);
s=s+b;
c++;
}
```

```
}
av=s/c;
return av;
}}
```

Main

```
import java.util.*;
public class Main
{
public static void main(String args[])
{
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
HashMap<Integer,Integer> h1=new HashMap<Integer,Integer>();
for(int i=0;i<n;i++)
{
h1.put(sc.nextInt(),sc.nextInt());
}
System.out.println(UserMainCode.getAverageOfOdd(h1));
sc.close();
}
}.
```

```
4
2
34
1
4
5
12
4
22
8
```

2.1.Common Elements(code 26)

Write a program to find out sum of common elements in given two arrays. If no common elements are found print - “No common elements”.
Include a class UserMainCode with a static method getSumOfIntersection which accepts
two integer arrays and their sizes. The return type (integer) should return the sum of common elements.
Create a Class Main which would be used to accept 2 Input arrays and call the static method present in UserMainCode.
Input and Output Format:
Input consists of 2+m+n integers. The first integer corresponds to m (Size of the 1st array), the second integer corresponds to n (Size of the 2nd array), followed by m+n integers corresponding to the array elements.
Output consists of a single Integer corresponds to the sum of common elements or a string “No common elements”.
Refer sample output for formatting specifications.
Assume the common element appears only once in each array.
Sample Input 1:
4
3
2
3
5
1
1
3
9
Sample Output 1:
4
Sample Input 2:
4
3
2
3
5
1
12
31
9
Sample Output 2:
No common elements

UserMainCode

```
public class UserMainCode {
public static int getSumOfIntersection(int a[],int b[],int n,int m)
{
int sum=0;
```

```
for(int i=0;i<a.length;i++)
{
for(int j=0;j<b.length;j++)
{if(a[i]==b[j])
sum=sum+a[i];
}}
if(sum==0)
return -1;
else
return sum;
}
}
```

Main

```
import java.util.Scanner;
public class Main {
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
int m=sc.nextInt();
int[] a=new int[n];
int[] b=new int[m];
for(int i=0;i<n;i++){
a[i]=sc.nextInt();}
for(int i=0;i<m;i++){
b[i]=sc.nextInt();}
int u=UserMainCode.getSumOfIntersection (a,b,n,m);
if(u==-1)
System.out.println("No common elements");
else
System.out.println(u);
sc.close();
}}
```

```
4
3
2
3
5
1
1
3
9
4
.
```

2.2.Common Elements(code 57)

Write a program to read two integer arrays and find the sum of common elements in both the arrays. If there are no common elements return -1 as output Include a class UserMainCode with a static method sumCommonElements which accepts two single integer array. The return type (integer) should be the sum of common elements.
Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode. Assume that all the elements will be distinct. Input and Output Format: Input consists of 2n+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, The last n elements correspond to the elements of the second array. Output consists of a single Integer value. Refer sample output for formatting specifications.
Assume that the maximum number of elements in the array is 20.
Sample Input 1:

```
4
1
2
3
4
2
3
6
7
```

Sample Output 1:

```
5
```

UserMainCode

```
public class UserMainCode {
	public static int sumCommonElements(int a[],int b[]){
		int sum = 0 ;
		for(int i=0;i<a.length;i++)
		{
		for(int j=0;j<b.length;j++){
		if(a[i]==b[j])
		sum = sum + a[i];}
		}
		if(sum==0)
		return -1;
		else return sum;
		}
		}
```

Main

```
package CommonElements;
import java.util.*;
public class Main {
private static Scanner s ;
;
public static void main(String[] args) {
s = new Scanner (System.in);
int n = s.nextInt();
int a[] = new int[n];
int b[] = new int[n];
for(int i=0;i<n;i++)
{
a[i] = s.nextInt();
```

```
}
for(int i=0;i<n;i++)
{
b[i] = s.nextInt();
}
System.out.println(UserMainCode.sumCommonElements(a, b));
}
}
4
1
2
3
4
2
3
6
7
5
```

3.Generate the series

Given a method taking an odd positive Integer number as input. Write code to evaluate the following series:
1+3-5+7-9…+/-n.
Include a class UserMainCode with a static method addSeries which accepts a positive integer .
The return type of the output should be an integer .
Create a class Main which would get the input as a positive integer and call the static method addSeriespresent in the UserMainCode.
Input and Output Format:
Input consists of a positive integer n.
Output is a single integer .
Refer sample output for formatting specifications.
Sample Input 1:
9
Sample Output 1:
-3
Sample Input 2:
11
Sample Output 2:
8

1.Method Name- addSeries
UserMainCode class

```
import java.util.ArrayList;
import java.util.List;
public class UserMainCode {
 public static int addSeries(int n){
 List<Integer> l1=new ArrayList<Integer>();
 for(int i=1;i<=n;i++)
 if(i%2!=0)
 l1.add(i);
 int n1=l1.get(0);
 for(int i=1;i<l1.size();i++)
 if(i%2!=0)
 n1=n1+l1.get(i);
 else
 n1=n1-l1.get(i);
 return n1;
 }
}
```

Main class

```
import java.util.*;
 public class Main {
 public static void main(String[] args) {
 Scanner s=new Scanner(System.in);
```

```
 int n=s.nextInt();
 System.out.println(UserMainCode.addSeries(n));
 s.close();
 }
}
```

2.Method Name- consecutiveSumSubofOddNos

UserMainCode

```
import java.util.*;
public class UserMainCode {
public static int consecutiveSumSubofOddNos(int n){
List<Integer> l1=new ArrayList<Integer>();
for(int i=1;i<=n;i++)
if(i%2!=0)
l1.add(i);
int n1=l1.get(0);
for(int i=1;i<l1.size();i++)
if(i%2!=0)
n1=n1+l1.get(i);
else
n1=n1-l1.get(i);
return n1;
}
}
```

Main class

```
import java.util.Scanner;
public class Main{
public static void main(String[] args) {
Scanner s=new Scanner(System.in);
int n=s.nextInt();
System.out.println(UserMainCode.consecutiveSumSubofOddNos(n));
}
}
```

```
11
8
```

4.Palindrome in range

Write a program to input two integers, which corresponds to the lower limit and upper limit respectively, and find the sum of all palindrome numbers present in the range including the two numbers. Print the sum.
Include a class UserMainCode with a static method addPalindromes which accepts two integers. The return type (Integer) should return the sum if the palindromes are present, else return 0.
Create a Class Main which would be used to accept two integer and call the static method present in UserMainCode.
Note1 : A palindrome number is a number which remains same after reversing its digits.
Note2 : A single digit number is not considered as palindrome.
Input and Output Format:
Input consists of 2 integers, which corresponds to the lower limit and upper limit respectively.
Output consists of an Integer (sum of palindromes).
Refer sample output for formatting specifications.
Sample Input :
130
150
Sample Output :
272
(131+141 = 272)

UserMainCode

```
public class UserMainCode {
public static int addPalindromes(int n1,int n2){
int sum=0;
for(int i=n1;i<=n2;i++){
int r=0,n3=i;
while(n3!=0){
r=(r*10)+(n3%10);
n3=n3/10;
}
if(r==i)
sum=sum+i;
}
return sum;
}
}
```

Main

```
import java.util.Scanner;
public class Main {
public static void main(String[] args) {
Scanner s=new Scanner(System.in);
int n1=s.nextInt();
int n2=s.nextInt();
System.out.println(UserMainCode.addPalindromes(n1,n2));
s.close();
}
}
```

```
130
150
272
```

.5.Find Digits

For a given double number with atleast one decimal value, Write a program to compute the number of digits before and after the decimal point in the following format – noOfDigitsBeforeDecimal:noOfDigitsAfterDecimal.
Note: Ignore zeroes at the end of the decimal (Except if zero is the only digit after decimal. Refer Example 2 and 3)
Include a class UserMainCode with a static method findNoDigits which accepts the decimal
value. The return type is string.
Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.
Input and Output Format:
Input consists of a double.
Output consists of string.
Refer sample output for formatting specifications.
Sample Input 1:
843.21
Sample Output 1:
3:2
Sample Input 2:
20.130
Sample Output 2:
2:2
Sample Input 3:
20.130

UserMainCode

```
import java.util.StringTokenizer;
public class UserMainCode {
public static String findNoDigits(double d) {
int n1=0,n2=0;
String s=String.valueOf(d);
StringTokenizer t=new StringTokenizer(s,".");
String s1=t.nextToken();
String s2=t.nextToken();
n1=s1.length();
n2=s2.length();
if(s1.charAt(0)=='0')
n1=s1.length()-1;
if(n2!=1)
if(s2.charAt(s2.length()-1)=='0')
n2=s2.length()-1;
String s3=String.valueOf(n1)+":"+String.valueOf(n2);
return s3;
}
}
```

Main

```
import java.util.*;
public class Main {
```

```
public static void main(String[] args) {
Scanner s=new Scanner(System.in);
double d=s.nextDouble();
System.out.println(UserMainCode.findNoDigits(d));
}}
```

```
20.130
2:2
```

.6.Cube and Square

Write a program to get an int array as input and identify even and odd numbers. If number is odd get cube of it, if number is even get square of it. Finally add all cubes and squares together and return it as output.
Include a class UserMainCode with a static method addEvenOdd which accepts integer array as input.
The return type of the output is an integer which is the sum of cubes and squares of elements in the array.
Create a class Main which would get the input and call the static method addEvenOdd present in the UserMainCode.
Input and Output Format:
Input consists of integer array.
Output is an integer sum.
Refer sample output for formatting specifications.
Sample Input 1:
5
2
6
3
4
5
Sample Output 1:
208

UserMainCode

```
public class UserMainCode {
public static int addEvenOdd(int[] a) {
int n1=0,n2=0;
for(int i=0;i<a.length;i++)
if(a[i]%2==0)
n1+=(a[i]*a[i]);
else
n2+=(a[i]*a[i]*a[i]);
return n1+n2;
}
}
```

Main

```
import java.util.Scanner;
public class Main {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
int a[]=new int[n];
for(int i=0;i<n;i++){
a[i]=sc.nextInt();
}
System.out.println(UserMainCode.addEvenOdd(a));
sc.close();
}
}
5
2
6
3
```

4
5
208