

WEEK-3

ROLL NO:230701235

1) Given an integer array as input, perform the following operations on the array, in the below specified sequence.

1. Find the maximum number in the array.
2. Subtract the maximum number from each element of the array.
3. Multiply the maximum number (found in step 1) to each element of the resultant array.

After the operations are done, return the resultant array.

Example 1: input1 = 4 (represents the number of elements in the

input1 array) input2 = {1, 5, 6, 9}

Expected Output = {-72, -36, 27, 0}

For example:

Input	Result
4 1 5 6 9	-72 -36 -27 0
5 10 87 63 42 2	-6699 0 -2088 -3915 -7395
2 -9 9	-162 0

CODE:

```
import java.util.*; public class main{
public static void main(String [] args){
int n;

Scanner s=new Scanner(System.in);

n=s.nextInt();

int[] a=new int[n];

for(int i=0;i<n;i++){

a[i]=s.nextInt();
```

```

    }
    s.nextLine();
int max=0;    for(int
i=0;i<n;i++){
if(max<a[i]){
max=a[i];

    }
}

for(int i=0;i<n;i++){
a[i]-=max;    a[i]*=max;

    System.out.print(a[i]+" ");

}

```

OUTPUT:

	Input	Expected	Got	
✓	4 1 5 6 9	-72 -36 -27 0	-72 -36 -27 0	✓
✓	5 10 87 63 42 2	-6699 0 -2088 -3915 -7395	-6699 0 -2088 -3915 -7395	✓
✓	2 -9 9	-162 0	-162 0	✓

Passed all tests! ✓

2) You are provided with a set of numbers (array of numbers).

You have to generate the sum of specific numbers based on its position in the array set provided to you.

This is explained below:

Example 1:

Let us assume the encoded set of numbers given to you is:

input1:5 and input2: {1, 51, 436, 7860, 41236} Step

1:

Starting from the 0th index of the array pick up digits as per below:

0th index – pick up the units value of the number (in this case is 1).

1st index - pick up the tens value of the number (in this case it is 5).

2nd index - pick up the hundreds value of the number (in this case it is 4).

3rd index - pick up the thousands value of the number (in this case it is 7).

4th index - pick up the ten thousands value of the number (in this case it is 4).

(Continue this for all the elements of the input array).

The array generated from Step 1 will then be – {1, 5, 4, 7, 4}.

Step 2:

Square each number present in the array generated in Step 1.

{1, 25, 16, 49, 16} Step

3:

Calculate the sum of all elements of the array generated in Step 2 to get the final result. The result will be = 107.

Input	Result
5 1 51 436 7860 41236	107
5 1 5 423 310 61540	53

CODE: import java.util.*;

public class main{ public static void

main(String [] args){

 int n;

 Scanner s=new Scanner(System.in);

 n=s.nextInt();

int[] a=new int [n];

```

for(int i=0;i<n;i++){
a[i]=s.nextInt();

    }
    int temp=0;    int sum=0;
for(int i=0;i<n;i++){
double c=i;    double t=10;
double b=Math.pow(t,c);
    if(b>=a[i]){
temp=0;
        }    else{
temp=a[i];
temp=temp/(int)b;

        }
        sum+=temp*temp;
    }
    System.out.print(sum+1);

}
}

```

OUTPUT:

	Input	Expected	Got	
✓	5 1 51 436 7860 41236	107	107	✓
✓	5 1 5 423 310 61540	53	53	✓

Passed all tests! ✓

3) Given an array of numbers, you are expected to return the sum of the longest sequence of POSITIVE numbers in the array.

If there are NO positive numbers in the array, you are expected to return -1.

In this question's scope, the number 0 should be considered as positive.

Note: If there are more than one group of elements in the array having the longest sequence of POSITIVE numbers, you are expected to return the total sum of all those POSITIVE numbers (see example 3 below).

input1 represents the number of elements in the array.

input2 represents the array of integers.

CODE:

```
import java.util.*; public class main{
public static void main(String [] args){
int n;

Scanner s= new Scanner(System.in);
n=s.nextInt();    int[] a=new int[n];
int upper=0;    int lower=0;

int i=0;    int
count1=0;
for(i=0;i<n;i++){
a[i]=s.nextInt();
if(a[i]<0){
count1+=1;
```

```

        }
    }
    i=0;
    int count=0;
    int initial=0;    int
    max=0;
    if(count1==n){
        System.out.print(-1);

    }
    else{
        while(i<n){
            count=0;
            initial=i;
            while(a[i]>=0){
                count++;
                i++;
            }
            if(max <count){
                upper=(initial-1)+count;
                lower=initial;        max=count;
            }
            i++;
        }
        int sum=0;
        while(lower<=upper){
            sum+=a[lower];        lower++;

```

```

    }
    System.out.print(sum);
}
}
}

```

OUTPUT;

Input	Result
16 -12 -16 12 18 18 14 -4 -12 -13 32 34 -5 66 78 78 -79	62
11 -22 -24 -16 -1 -17 -19 -37 -25 -19 -93 -61	-1
16 -58 32 26 92 -10 -4 12 0 12 -2 4 32 -9 -7 78 -79	174