

Date : 21.08.2024

Week 5 programs

1.

Question 1

Correct

Mark 1.00 out of 1.00

Given an array A of positive integers, let S be the sum of the digits of the minimal element of A.
Return 0 if S is odd, otherwise return 1.

Example 1:**Input:**

8

34 23 1 24 75 33 54 8

Output:

0

Explanation:The minimal element is 1, and the sum of those digits is $S = 1$ which is odd, so the answer is 0.**Example 2:****Input:**

5

99 77 33 66 55

Output:

1

Explanation:The minimal element is 33, and the sum of those digits is $S = 3 + 3 = 6$ which is even, so the answer is 1.**Constraints:**

- $1 \leq A.length \leq 100$
- $1 \leq A[i] \leq 100$

```
import java.util.* ;
```

```
class MinSum {  
    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();  
        }  
        int s = a[0];  
        for (int i = 0 ; i<n ; i++){  
            if (a[i] < s)  
                s = a[i];  
        }  
    }  
}
```

```
String w = String.valueOf(s);  
int z = 0;  
while (s>0){  
    z += s%10 ;  
    s = s/10;  
}
```

```

if (z%2 == 0 ){
System.out.print("1");}
else {
System.out.print("0");}
}
}

```

```
C:\Users\kkknf\OneDrive\Documents\Javacodes>javac MinSum.java
```

```
C:\Users\kkknf\OneDrive\Documents\Javacodes>java MinSum
```

```
8
```

```
34 23 1 24 75 33 54 8
```

```
0
```

```
C:\Users\kkknf\OneDrive\Documents\Javacodes>java MinSum
```

```
5
```

```
99 77 33 66 55
```

```
1
```

```
C:\Users\kkknf\OneDrive\Documents\Javacodes>
```

Question 2

Correct

Mark 1.00 out of 1.00

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 it 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.

Note:

1) While picking up a number in Step 1, if you observe that the number is smaller than the required position then use 0.

2) In the given function, input1[] is the array of numbers and input2 represents the number of elements in input1.

Example 2:

input1: 5 and input2: {1, 5, 423, 310, 61540}

Step 1:

Generating the new array based on position, we get the below array:

{1, 0, 4, 0, 0}

In this case, the value in input1 at index 1 and 3 is less than the value required to be picked up based on position, so we use a 0.

Step 2:

{1, 0, 16, 0, 16}

Step 3:

The final result = 53.

For example:

Input	Result
5	107
5 51 436 7860 41236	
5	53

```
import java.util.* ;
```

```
class SumPosition {  
    public static void main (String args []){  
        Scanner sc = new Scanner(System.in);  
        int n = sc.nextInt();  
        int[] a = new int[n];  
        int[] b = new int[n];  
        for (int i = 0 ; i<n ; i++){  
            a[i] = sc.nextInt();  
        }  
        int x=0,y=0;  
        for (int i = 0 ; i < n ; i++)  
        {  
            y = (int) Math.pow(10,i+1);  
            x = (a[i] / (y/10)) % 10;  
            b[i] = x*x ;  
        }  
        int sum =0 ;  
        for (int i = 0 ; i<n ; i++){  
            sum += b[i];  
        }  
        System.out.print(sum);  
    }  
}
```

```
PS C:\Users\kkknf\OneDrive\Documents\Javacodes> javac SumPosition.java  
PS C:\Users\kkknf\OneDrive\Documents\Javacodes> java SumPosition  
5  
1 5 423 310 61540  
53  
PS C:\Users\kkknf\OneDrive\Documents\Javacodes> |
```

Question 3
Correct
Mark 1.00 out of 1.00

The program must accept N integers and an integer K as the input. The program must print every K integers in descending order as the output.

Note: If $N \% K \neq 0$, then sort the final $N\%K$ integers in descending order.

Boundary Condition(s):
 $1 \leq N \leq 10^4$
 $-9999 \leq \text{Array Element Value} \leq 9999$

Input Format:
The first line contains the values of N and K separated by a space.
The second line contains N integers separated by space(s).

Output Format:
The first line contains N integers.

Example Input/Output 1:

Input:
7 3
48 541 23 68 13 41 6

Output:
541 48 23 68 41 13 6

Explanation:
The first three integers are 48 541 23, after sorting in descending order the integers are **541 48 23**.
The second three integers are 68 13 41, after sorting in descending order the integers are **68 41 13**.
The last integer is **6**.
The integers are **541 48 23 68 41 13 6**.
Hence the output is **541 48 23 68 41 13 6**.

```
import java.util.* ;
```

```
class DividenSort {
```

```
public static void main (String args []){
```

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

```
int n = sc.nextInt();
```

```
int k = sc.nextInt();
```

```
int[] a = new int[n];
```

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

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

```
}
```

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

```
int end = Math.min(i + k, n);
```

```
for (int j = i; j < end - 1; j++) {  
    for (int l = j + 1; l < end; l++) {  
        if (a[j] < a[l]) {  
            int temp = a[j];  
            a[j] = a[l];  
            a[l] = temp;  
        }  
    }  
}  
  
for (int i = 0; i < n; i++) {  
    System.out.print(a[i] + " ");  
}  
sc.close();  
}  
}
```

```
C:\Users\kkknf\OneDrive\Documents\Javacodes>javac DividenSort.java  
C:\Users\kkknf\OneDrive\Documents\Javacodes>java DividenSort  
7 3  
48 541 23 68 13 41 6  
541 48 23 68 41 13 6  
C:\Users\kkknf\OneDrive\Documents\Javacodes>
```