

Week 5 sample program**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:

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

Program:

```
import java.util.Scanner; public class
MinElementDigitSum {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements in the array: ");
        int size = scanner.nextInt();    if (size <= 0) {
            System.out.println("Array size must be greater than zero.");
            return;
        }
        int[] array = new int[size];
```

```
System.out.println("Enter the elements of the array:");
for (int i = 0; i < size; i++) {
array[i] = scanner.nextInt();
}
int minElement = array[0];
for (int i = 1; i < size; i++) {    if
(array[i] < minElement) {
minElement = array[i];
}
}
int sumOfDigits = 0;    int
number = minElement;    while
(number > 0) {    sumOfDigits
+= number % 10;
    number /= 10;
}
System.out.println(sumOfDigits % 2 == 0 ? 1 : 0);
}
}
```

Output:

```
Enter the number of elements in the array: 8
```

```
Enter the elements of the array:
```

```
34
```

```
23
```

```
1
```

```
24
```

```
75
```

```
33
```

```
54
```

```
8
```

```
0
```

```
Enter the number of elements in the array: 5
```

```
Enter the elements of the array:
```

```
99
```

```
77
```

```
33
```

```
66
```

```
55
```

```
1
```

2.

Question 2

Closed

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 input2 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, 0}

Step 3:

The final result = 17.

For example:

Input	Result
5	107
{ 1, 51, 436, 7860, 41236 }	
5	53

Program:

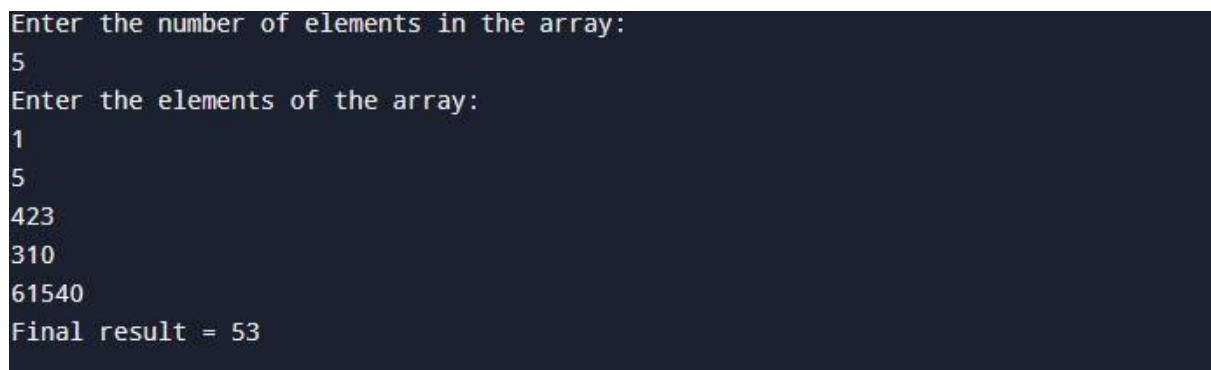
```
import java.util.Scanner; public class
DigitSumCalculator {    public static void
main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter the number of elements in the array:");
    int n = scanner.nextInt();    int[] input = new int[n];
    System.out.println("Enter the elements of the array:");
    for (int i = 0; i < n; i++) {
        input[i] = scanner.nextInt();
    }
}
```

```
int finalSum = calculateFinalSum(input);

System.out.println("Final result = " + finalSum);
}

public static int calculateFinalSum(int[] input) {
int finalSum = 0;    for (int i = 0; i < input.length;
i++) {    int currentNumber =
input[i];    int digitPosition = i + 1;    int digit =
getDigitAtPosition(currentNumber, digitPosition);    finalSum
+= digit * digit;
    }
    return finalSum;
}

public static int getDigitAtPosition(int number, int position) {    String
numberStr = Integer.toString(number);    int length =
numberStr.length();    if (length < position) {
    return 0;
    }
    char digitChar = numberStr.charAt(length - position);
return Character.getNumericValue(digitChar);
}
}
```

Output:A screenshot of a terminal window with a dark background and light-colored text. It shows the execution of a Java program. The first prompt is "Enter the number of elements in the array:", followed by the input "5". The second prompt is "Enter the elements of the array:", followed by the inputs "1", "5", "423", "310", and "61540" on separate lines. The final output line is "Final result = 53".

```
Enter the number of elements in the array:
5
Enter the elements of the array:
1
5
423
310
61540
Final result = 53
```

3.

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): $1 \leq N \leq 10^4$ $-99999 \leq \text{Array Element Value} \leq 99999$ **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**.

Program:

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

```
Scanner scanner = new Scanner(System.in);
System.out.println("Enter the values of N and K:");
int N = scanner.nextInt();    int K = scanner.nextInt();

    System.out.println("Enter the " + N + " elements:");    int[]
arr = new int[N];    for (int i = 0; i < N; i++) {
arr[i] = scanner.nextInt();
    }
    for (int i = 0; i < N; i += K) {
int end = Math.min(i + K, N);
Arrays.sort(arr, i, end);    reverse(arr,
i, end - 1);
    }
    for (int num : arr) {
        System.out.print(num + " ");
    }
}

public static void reverse(int[] arr, int start, int end) {
while (start < end) {    int temp = arr[start];    arr[start]
= arr[end];    arr[end] = temp;
    start++;
end--;
    }
}
}
```

Output:

```
Enter the values of N and K:
```

```
7 3
```

```
Enter the 7 elements:
```

```
48
```

```
541
```

```
23
```

```
68
```

```
13
```

```
41
```

```
6
```

```
541 48 23 68 41 13 6
```