Given a number as an integer input. Check which digit occurs the **maximum** number of times. Print that digit.

Input Format

Input contains an integer input

Constraints

1 <= integer <= 10^9

Output Format

print diigt with highest freq

Sample Input 0

```
11234
```

Sample Output 0

```
1
```

Submitted Code

Language: Java 15

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

public class Solution {

    public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int frq [] = new int[10];

        while(n>0){
            int rem = n%10;
            frq[rem]++;
            n/=10;
        }
        int max = Integer.MIN_VALUE;
        int ans =0;
        for(int i=0;i<frq.length;i++){
            if(max<frq[i]){
                max=frq[i];
                ans =i;
        }}
        System.out.print(ans);
    }
}
```

Given an array **arr** of integers arr, *a lucky integer is an integer that has a frequency in the array equal to its value*. Return the **largest** lucky integer in the array. If there is no lucky integer return **-1**.

Input Format

- N as size
- N int value as array elements

Constraints

- 1<=N<=10^5
- 1<=arr[i]<10

Output Format

Lucky Number

Sample Input 0

```
5
1 2 2 3 4
```

Sample Output 0

```
2
```

Explanation 0

There are two lucky numbers:

1 because frequency of 1 is equals to 1

2 because frequency of 2 is equals to 2.

2 is largest among the two lucky numbers, hence output is 2.

Submitted Code

Language: Java 15

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

public class Solution {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        int[] arr = new int[n];
        for(int i=0;i<n;i++)arr[i]=sc.nextInt();

        int [] frq = new int[10];
        for(int i =0;i<n;i++){
            frq[arr[i]]++;
        }

        int max = -1;
        for(int i =1;i<frq.length;i++){
            if(frq[i]==i){
               max= Math.max(max,i);
            }
        }
        System.out.print(max);
    }
}
```

Samantha is a Scrabble champion who always manages to find the best **anagrams** from her letters. One day, her friend Jake asked her how she does it so quickly. Samantha revealed her secret: a Java function that she wrote to determine if two strings are anagrams of each other.

take two strings, **s** and **t**, and compare the frequency of each character in both strings. If the frequency of each character is the same in both strings, then they are anagrams of each other.

Input Format

First line contain string **S**.

Second line contain string **T**.

Constraints

```
1<=S.length() , T.length()<=10^5
```

Output Format

Return Yes or No

Sample Input 0

```
anagram
nagaram
```

Sample Output 0

```
Yes
```

Submitted Code

Language: Java 15

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

public class Solution {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s1 = sc.next();
        String s2 = sc.next();

        int frq[] = new int[26];
        for(int i=0;i<s1.length();i++)frq[s1.charAt(i)- 'a']++;
        for(int i=0;i<s2.length();i++)frq[s2.charAt(i)- 'a']--;

        for(int i=0;i<frq.length;i++){
            if(frq[i]!=0){
                System.out.print("No");
                return;
            }
        }
        System.out.print("Yes");
    }
}
```

Submitted Code

Language: Java 15

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

public class Solution {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s1 = sc.next();
        String s2 = sc.next();
        if(s1.length()!=s2.length()){
            System.out.print("No");
            return;
        }
        int frq1[] = new int[26];
        for(int i=0;i<s1.length();i++)frq1[s1.charAt(i)- 'a']++;
        int frq2[] = new int[26];
        for(int i=0;i<s2.length();i++)frq2[s2.charAt(i)- 'a']++;

        for(int i=0;i<frq1.length;i++){
            if(frq1[i]!=frq2[i]){
                System.out.print("No");
                return;
            }
        }
        System.out.print("Yes");
    }
}
```

Jack was a treasure hunter who spent his life traveling the world, looking for hidden riches. One day, while exploring a remote island, he stumbled upon an old map that promised great wealth.

The map showed a series of clues that led to a hidden treasure trove, but it was written in a secret code that Jack could not decipher. He knew that he needed to find a way to decode the map if he was going to find the treasure.

Jack remembered a Java program he had learned about that could find an odd occurring element in an array of integers. He realized that he could use this program to decode the map.

Given an array `arr[]` of **N** integers, find a number which occurred odd times in the array. If such an element exists, print that respective element, else print There is no odd occurring element.

Note :- Consider there is only one element which occurs odd time.

Hint use frequency array.

Input Format

The first line contains **N**, i.e. the size of the array.

The second line contains **N** space-separated positive integers `arr[i]` denoting elements of the array.

Constraints

```
0 <= N <= 10^4

0 <= arr[i] <= 9
```

Sample Input 0

```
5
1 1 1 2 2
```

Sample Output 0

```
1
```

Submitted Code

Language: Java 15

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

public class Solution {

    public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
       int n = sc.nextInt();
        int arr[] = new int[n];
        for(int i=0;i<n;i++)arr[i]=sc.nextInt();

       int frq [] = new int[10];
       for(int i=0;i<n;i++)frq[arr[i]]++;

       for(int i=0;i<frq.length;i++){
           if(frq[i]%2!=0){
               System.out.print(i);
               return;
       }}
       System.out.print("There is no odd occurring element");
    }
}
```

Given an integer array `nums` and an integer **k**, return the **k** most frequent elements.

If 2 elements have same frequency than print them in decreasing order.

Input Format

The first line contains **N**, i.e. the size of the array.

The second line contains **N** space-separated positive integers `nums[i]` denoting elements of the array.

The third line contains integer **k**.

Constraints

```
1 <= N <= 10^5

0 <= nums[i] <= 9

k is in the range [1, the number of unique elements in the array].

It is guaranteed that the answer is unique.
```

Output Format

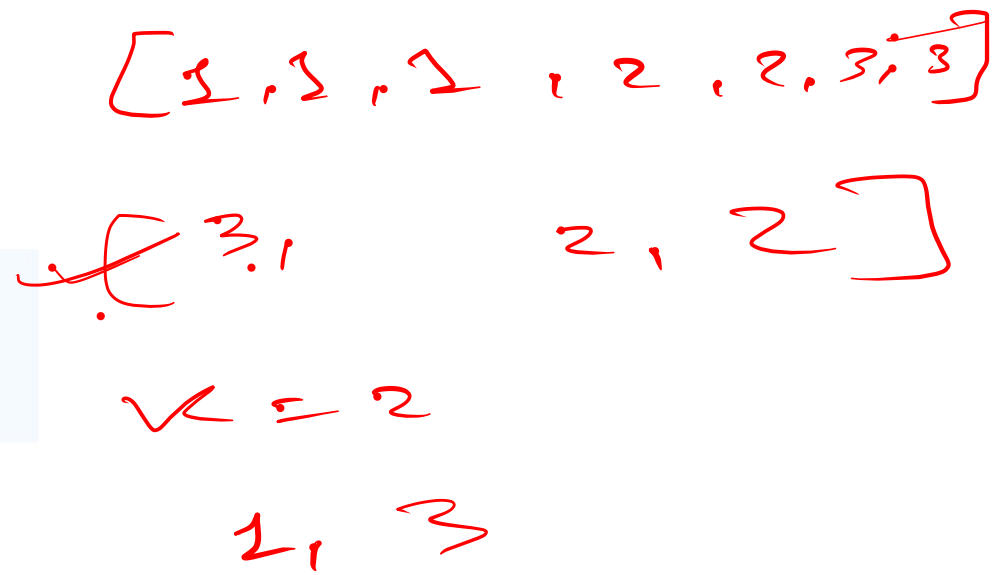
Return the top k frequent elements.

Sample Input 0

```
7
1 1 1 2 2 3 3
2
```

Sample Output 0

```
1 3
```



Submitted Code

Language: Java 15

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

public class Solution {

    public static void main(String[] args) {
         Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();

        int[] arr = new int[n];
        for(int i=0;i<n;i++)arr[i]=sc.nextInt();

        int k = sc.nextInt();
        int [] frq = new int[10];
        for(int i =0;i<n;i++){
            frq[arr[i]]++;
        }

        for(int i=0;i<k;i++){
            int max = Integer.MIN_VALUE,idx=0;
            for(int j=0;j<frq.length;j++){
                if(max<=frq[j]){
                    max=frq[j];
                    idx=j;
                }}
                System.out.print(idx+" ");
                frq[idx]=-1;
            }

    }
}
```