

Matrix

String

String Concat

Two
Arrays

Reverse

String

str = "Sagen"

rev = ""

1

for(int i = str.length()-1; i >= 0; i--)

{
 sysout

rev += str.charAt(i);

}

rev

Diagram illustrating matrix reversal logic:

Initial Matrix (3x3):

1	2	3
4	5	6
7	8	9

Reversed Matrix (3x3):

3	2	1
6	5	4
9	8	7

Code Snippet:

```
System.out.println("REV");  
public static void rotate(int[][] mat, int row, int col, int temp) {  
    int temp = mat[row][col];  
    mat[row][col] = mat[row][col-1];  
    mat[row][col-1] = temp;  
}
```

Logic:

- int temp = mat[row][col];
- temp = mat[row][col-1];
- mat[row][col] = temp;

Given two strings str & target, return the index where target string occurs for the first time in String str.

Input Format

- The first line contains the string str.
- The second line contains the string target.

Constraints

- $1 \leq \text{str.length} \leq 10^4$
- str consists of lowercase English letters.

Output Format

- Print the index where the target string occurs for the first time.
- If the target string is not found, print -1.

Sample Input 0

```
geekster
st
```

Sample Output 0

```
4
```

Explanation 0

The string "geekster" contains the target string "st" from index [4-5]. So, the starting index (4) is printed as the result.

Sample Input 1

```
geekster
ab
```

Sample Output 1

```
-1
```

Explanation 1

String "geekster" does not contain "ab". So, result is -1.

geekster

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

ch = str.charAt(i);

ch = target.charAt(0);

ch = ch

for target

if false

flag = false

stop

geekster

2

contains

str.contains()

```

public static boolean
threeTimesOccur(int[] arr, int n) {

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

        for (int j = 0; j < n; j++) {
            if (arr[i] == arr[j]) {
                count++;
            }
        }

        if (count == 3) {
            return true;
        }
    }

    return false;
}

```

[1, 1, 1, 1, 2, 2]

for (int i = 0; i < n; i++) {
 int count = 0;
 for (int j = 0; j < n; j++) {
 if (arr[i] == arr[j]) {
 count++;
 }
 }
 if (count == 3) {
 return true;
 }
 }
 return false;

Sub 1E

[1, 1, 1, 1, 2, 2]

i = 0
 i = 1
 i = 2
 i = 3
 i = 4
 i = 5

HashMap

Given an array of integers **arr**, replace each element with its rank.

The rank represents how large the element is. The rank has the following rules:

- Rank is an integer starting from 1.
- The larger the element, the larger the rank. If two elements are equal, their rank must be the same.
- Rank should be as small as possible.

Input Format

First line contains an integer **n**.

Second line contains an array of integers of size **n**.

Constraints

$0 \leq \text{arr.length} \leq 10^5$

$10^9 \leq \text{arr}[i] \leq 10^9$

Output Format

Returns an array.

Sample Input 0

```
4
40 10 20 30
```

Sample Output 0

```
4 1 2 3
```

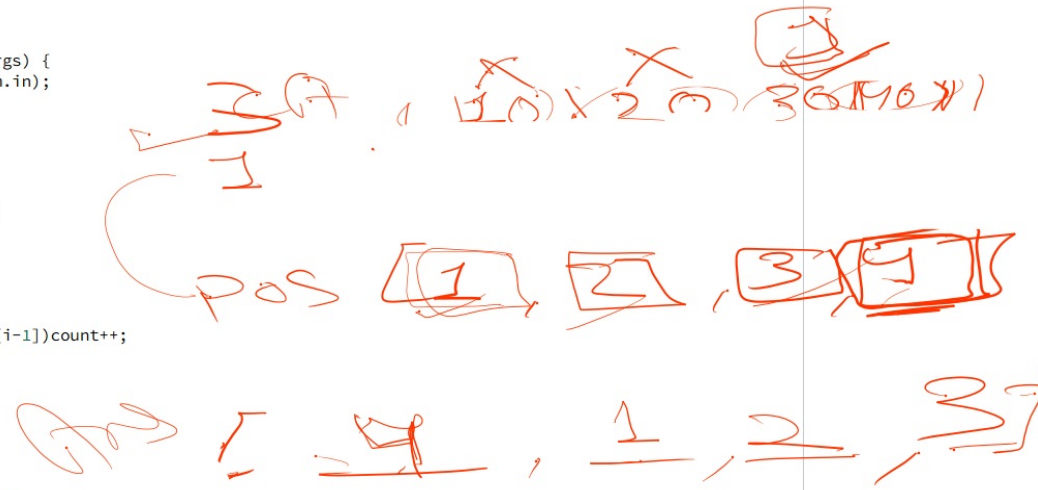
Explanation 0

40 is the largest element. 10 is the smallest. 20 is the second smallest. 30 is the third smallest.

20 is the second smallest. 30 is the third smallest.



```
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 copyArray [] = arr.clone();
    Arrays.sort(copyArray);
    int pos[] = new int[n];
    int count = 1;
    pos[0] = count;
    for(int i = 1; i < n; i++){
        if(copyArray[i] != copyArray[i-1]) count++;
        pos[i] = count;
    }
    int ans [] = new int[n];
    for(int i = 0; i < n; i++){
        int idx = 0;
        for(int j = 0; j < n; j++){
            if(arr[i] == copyArray[j]){
                idx = j;
                break;
            }
        }
        ans[i] = pos[idx];
    }
    for(int i = 0; i < n; i++){
        System.out.print(ans[i] + " ");
    }
}
```



```
1 import java.io.*;
2 import java.util.*;
3
4 public class Solution {
5
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         int n = sc.nextInt();
9         int arr[] = new int[n];
10        for(int i = 0; i < n; i++){
11            arr[i] = sc.nextInt();
12        }
13        int copyArray [] = arr.clone();
14        Arrays.sort(copyArray);
15        int pos[] = new int[n];
16        int count = 1;
17        pos[0] = count;
18        for(int i = 1; i < n; i++){
19            if(copyArray[i] != copyArray[i-1]) count++;
20            pos[i] = count;
21        }
22
23        int ans [] = new int[n];
24        for(int i = 0; i < n; i++){
25            int idx = 0;
26            for(int j = 0; j < n; j++){
27                if(arr[i] == copyArray[j]){
28                    idx = j;
29                    break;
30                }
31            }
32            ans[i] = pos[idx];
33        }
34
35        for(int i = 0; i < n; i++){
36
37            System.out.print(ans[i] + " ");
38        }
39    }
```

A palindrome is a term that can be read the same forwards or backwards. To determine if a string is a palindrome, the string is converted to lowercase, and all non-alphanumeric characters are removed.

Input Format

string str as an input.

Constraints

$1 \leq \text{str.length} \leq 2 * 10^5$

str consists only of printable ASCII characters.

Output Format

return true or false.

Sample Input 0

A man, a plan, a canal: Panama

Sample Output 0

true

```
1 import java.io.*;
2 import java.util.*;
3
4 public class Solution {
5
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         String str = sc.nextLine();
9
10        str = str.toLowerCase().replaceAll("[^a-z0-9]", "");
11        int i = 0;
12        int j = str.length() - 1;
13        while(i <= j){
14            if(str.charAt(i) != str.charAt(j)){
15                System.out.print(false);
16                return;
17            }
18            i++;
19            j--;
20        }
21        System.out.print(true);
22    }
23 }
24 }
```



panama

amanaplanacanalpanama

amanaplanacanalpanama

palindrome

str

$\text{len} \leq 2$

()



```
1 import java.io.*;
2 import java.util.*;
3
4 public class Solution {
5
6     public static void main(String[] args) {
7         Scanner sc = new Scanner(System.in);
8         String str = sc.nextLine();
9
10        str = str.toLowerCase().replaceAll("[^a-z0-9]", "");
11
12        int i = 0;
13        int j = str.length() - 1;
14        while(i <= j){
15            if(str.charAt(i) != str.charAt(j)){
16                System.out.print(false);
17                return;
18            }
19            i++;
20            j--;
21        }
22
23        System.out.print(true);
24    }
25 }
```