**DSA Coding Problems**

**1.Bubble Sort:**

Input: arr[] = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

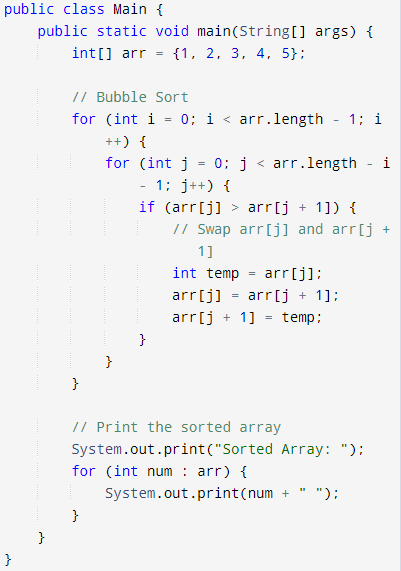
Output: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

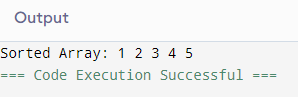
Input: arr[] = [1, 2, 3, 4, 5]

Output: [1, 2, 3, 4, 5]

Explanation: An array that is already sorted should remain unchanged after applying bubble sort.

Code:





**Time Complexity: O(n)**

**2.Quick Sort:**

Implement Quick Sort, a Divide and Conquer algorithm, to sort an array, **arr**[] in ascending order. Given an array, **arr**[], with starting index **low** and ending index **high**, complete the functions **partition()** and **quickSort()**. Use the last element as the pivot so that all elements less than or equal to the pivot come before it, and elements greater than the pivot follow it.

**Input:** arr[] = [4, 1, 3, 9, 7]

**Output:** [1, 3, 4, 7, 9]

**Explanation:** After sorting, all elements are arranged in ascending order.

**Input:** arr[] = [2, 1, 6, 10, 4, 1, 3, 9, 7]

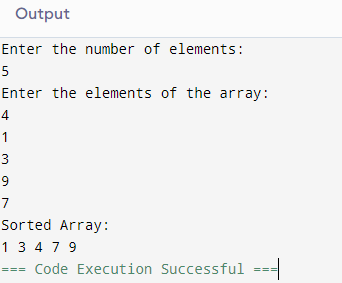
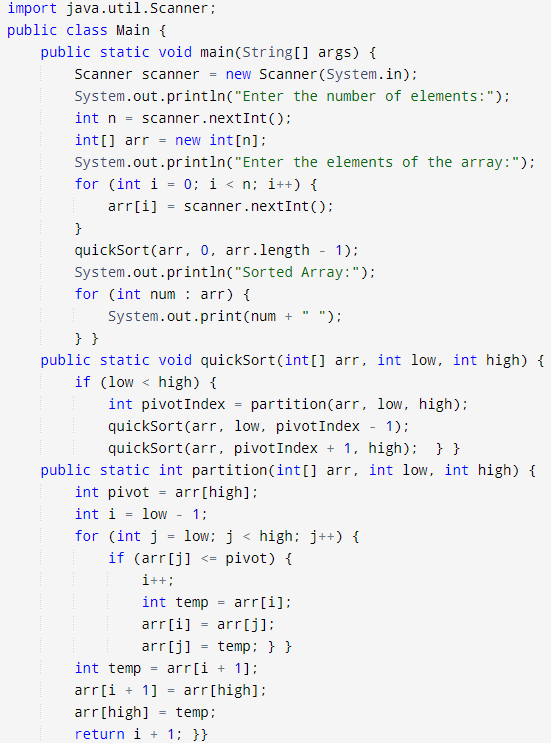
**Output: [**1, 1, 2, 3, 4, 6, 7, 9, 10]

**Explanation:** Duplicate elements (1) are retained in sorted order.

**Input:** arr[] = [5, 5, 5, 5]

**Output:** [5, 5, 5, 5]

**Explanation:** All elements are identical, so the array remains unchanged.



**Time Complexity: O(n)**

**3. Non Repeating Character:**

Given a string **s** consisting of **lowercase**Latin Letters. Return the first non-repeating character in **s**. If there is no non-repeating character, return **'$'.**

Note:When you return '$' driver code will output -1.

**Examples:**

**Input:** s = "geeksforgeeks"

**Output:** 'f'

**Explanation:** In the given string, 'f' is the first character in the string which does not repeat.

**Input:** s = "racecar"

**Output:** 'e'

**Explanation:** In the given string, 'e' is the only character in the string which does not repeat.

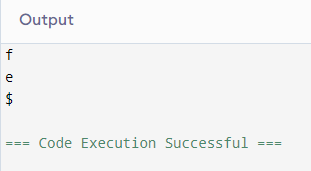
**Input:** s = "aabbccc"

**Output:** '$'

**Explanation:** All the characters in the given string are repeating.

Code:





**Time Complexity: O(n)**

**4.K Largest Element:**

Given an input stream **arr[]**of **n** integers. Find the **K**th largest element (not **K**th largest unique element) after insertion of each element in the stream and if the **Kth** largest element doesn't exist, the answer will be -1 for that insertion.  return a list of size n after all insertions.

**Example 1:**

**Input:**

**k =** 4, **n =** 6

**arr[] =** {1, 2, 3, 4, 5, 6}

**Output:**

-1 -1 -1 1 2 3

**Explanation:**

k = 4

For 1, the 4th largest element doesn't

exist so answer will be -1.

For 2, the 4th largest element doesn't

exist so answer will be -1.

For 3, the 4th largest element doesn't

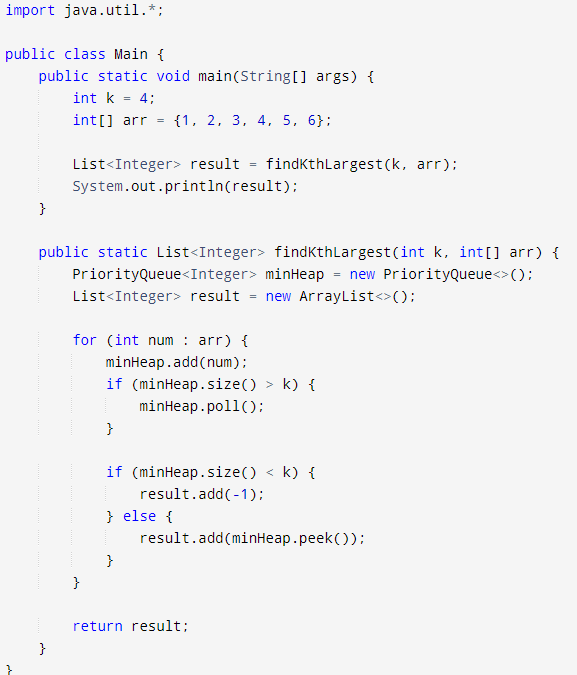
exist so answer will be -1.

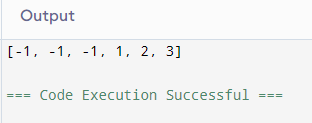
For 4, the 4th largest element is 1.

For 5, the 4th largest element is 2.

for 6, the 4th largest element is 3.

Code:





**Time Complexity: O(n**)

**5.Final the Largest Number:**

Given an integer **N** the task is to find the largest number which is smaller or equal to it and has its digits in non-decreasing order.

**Examples 1:**

**Input:**

N = 200

**Output:**

199

**Explanation:**

If the given number

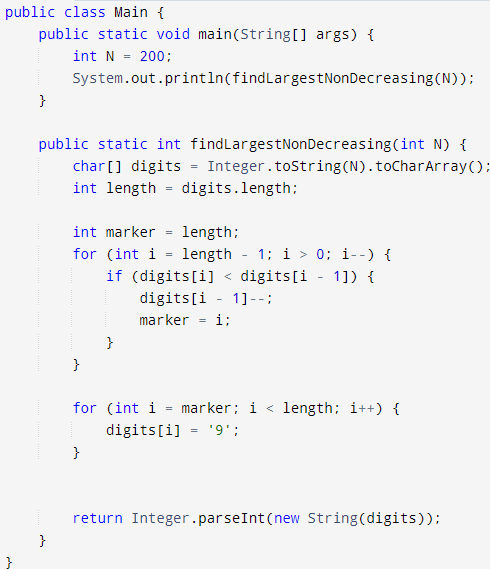
is 200, the largest

number which is smaller

or equal to it having

digits in non decreasing

order is 199.



**Time Complexity: O(n)**

