

Rajalakshmi Engineering College

Name: naveen prasath
Email: 240701352@rajalakshmi.edu.in
Roll no: 240701352
Phone: 9585322006
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Kavya, a software developer, is analyzing data trends. She has a list of integers and wants to identify the n th largest number in the list after sorting the array using QuickSort.

To optimize performance, Kavya is required to use QuickSort to sort the list before finding the n th largest number.

Input Format

The first line of input consists of an integer n , representing the size of the array.

The second line consists of n space-separated integers, representing the elements of the array `nums`.

The third line consists of an integer k , representing the position of the largest

number you need to print after sorting the array.

Output Format

The output prints the k-th largest number in the sorted array (sorted in ascending order).

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 6

-1 0 1 2 -1 -4

3

Output: 0

Answer

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int partition(int arr[], int low, int high) {
```

```
    int pivot = arr[high];
```

```
    int i = low - 1;
```

```
    for (int j = low; j < high; j++) {
```

```
        if (arr[j] < pivot) {
```

```
            i++;
```

```
            int temp = arr[i];
```

```
            arr[i] = arr[j];
```

```
            arr[j] = temp;
```

```
        }
```

```
    }
```

```
    int temp = arr[i + 1];
```

```
    arr[i + 1] = arr[high];
```

```
    arr[high] = temp;
```

```
    return i + 1;
```

```
}
```

```
void quickSort(int arr[], int low, int high) {
```

```

    if (low < high) {
        int pi = partition(arr, low, high);
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}

void findNthLargest(int* nums, int n, int k) {
    int target = n - k;
    int low = 0;
    int high = n - 1;

    while (low <= high) {
        int pivotIndex = partition(nums, low, high);

        if (pivotIndex == target) {
            printf("%d", nums[pivotIndex]);
            return;
        } else if (pivotIndex < target) {
            low = pivotIndex + 1;
        } else {
            high = pivotIndex - 1;
        }
    }
}

int main() {
    int n, k;
    scanf("%d", &n);
    int* nums = (int*)malloc(n * sizeof(int));
    for (int i = 0; i < n; i++) {
        scanf("%d", &nums[i]);
    }
    scanf("%d", &k);
    findNthLargest(nums, n, k);
    free(nums);
    return 0;
}

```

Status : Correct

Marks : 10/10