"Heaven's Light is Our Guide"

Department of Computer Science & Engineering

RAJSHAHI UNIVERSITY OF ENGINEERING & TECHNOLOGY

Lab Report

Course No: CSE 2202

Course Name: Sessional Based on CSE 2201

Submitted to:

Biprodip Pal

Assistant Professor
Department of Computer Science & Engineering

Submitted by:

Md. Al Siam
Department of Computer Science & Engineering Session:
2016-2017
Section: A
Roll No.: 1603008

Problem

Define a function that places the first element of its argument array in the k_{th} smallest position where it belongs on the sorted array.

Code

```
#include <bits/stdc++.h>
using namespace std;

int find_correct_position(int* ara, int target_pos, int last_pos){
    int pivot = ara[target_pos];
    int curr = last_pos+1;
    for(int i = last_pos; i >= 0; i--){
        if(ara[i] > pivot){
            curr--;
            swap(ara[i], ara[curr]);
        }
    }
    curr--;
    swap(ara[curr], ara[target_pos]);
    return curr;
}

int main(){
    int n;
    cin >> n; ///Number of data
    ///Entering elements:
    int ara[n+1];
    for(int i = 0; i < n; i++)
        cin >> ara[i];

    int target_pos;
    cin >> target_pos; ///Position of the element
```

```
                        ///that's to be placed correctly

    int temp = ara[target_pos];

    int ans = find_correct_position(ara, target_pos, n-1);

    cout << "The correct position for " << temp
        << " is " << ans << endl;

}
```

Sample Inputs and Outputs

Input	**Output**

10
21 20 46 81 11 6 9 17 111 44
0
	The correct position for 21 is 5

7
19 37 32 23 15 7 8
0
	The correct position for 19 is 3

10
21 20 46 81 11 6 9 17 111 44
3
	The correct position for 81 is 8

Problem

Use the concept to sort an array in divide and conquer approach by calling it recursively.

Code

```
#include <bits/stdc++.h>
using namespace std;

int find_correct_position(int* ara, int target_pos, int last_pos){
    int pivot = ara[target_pos];
    int curr = last_pos+1;
    for(int i = last_pos; i >= 0; i--){
        if(ara[i] > pivot){
            curr--;
            swap(ara[i], ara[curr]);
        }
    }
    curr--;
    swap(ara[curr], ara[target_pos]);
    return curr;
}

void my_sort(int* ara, int lo, int hi){
    if(lo >= hi) return;
    int curr_pos = find_correct_position(ara, lo, hi);
    my_sort(ara, lo, curr_pos-1);
    my_sort(ara, curr_pos+1, hi);
}

int main(){
    int n;
    cin >> n; ///Number of data

    ///Entering elements:
    int ara[n+1];
    for(int i = 0; i < n; i++)
        cin >> ara[i];

    my_sort(ara, 0, n-1);

    for(int i = 0; i < n; i++)
        cout << ara[i] << " ";
    cout << endl;
}
```

Sample Inputs & Outputs

Input	Output
10 21 20 46 81 11 6 9 17 111 44	6 9 11 17 20 21 44 46 81 111
7 19 37 32 23 15 7 8	7 8 15 19 23 32 37
11 11 10 99 8 7 55 4 32 2 10 69	2 4 7 8 10 10 11 32 55 69 99

Complexity Analysis

The `find_correct_position(int* ara, int target_pos, int last_pos)` function has only one for loop and some constant operations. Hence this function has complexity of **`O(n).`**

Hence, the overall complexity for the first problem is **`O(n).`**

The `my_sort(int* ara, int lo, int hi)` function has `find_correct_position()` of complexity O(n) and there is divide and conquer approach for sorting, having complexity of O(log n).

Hence, the overall complexity for the first problem is **`O(n log n).`**