Rajalakshmi Engineering College

Name: Alwin Abishek
Email: 241501017@rajalakshmi.edu.in
Roll no: 241501017
Phone: 9444177993
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 6_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are the lead developer of a text-processing application that assists writers in organizing their thoughts. One crucial feature is a character-sorting service that helps users highlight the most critical elements of their text.

To achieve this, you decide to enhance the service to sort characters in descending order using the Quick-Sort algorithm. Implement the algorithm to efficiently rearrange the characters, ensuring that it is sorted in descending order.

Input Format

The first line of the input consists of a positive integer value N, representing the number of characters to be sorted.

The second line of input consists of N space-separated lowercase alphabetical characters.

Output Format

The output displays the set of alphabetical characters, sorted in descending order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
a d g j k
Output: k j g d a

Answer

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

void swap(char* a, char* b) {
   char t = *a;
   *a = *b;
   *b = t;
}

// Function to partition the array for Quick Sort (descending order)
// arr[]: The array to be partitioned
// low: Starting index of the sub-array
// high: Ending index of the sub-array
int partition(char arr[], int low, int high) {
   char pivot = arr[high]; // Choosing the last element as the pivot
   int i = (low - 1);      // Index of smaller element

   for (int j = low; j <= high - 1; j++) {
      // If current element is greater than or equal to the pivot (for descending
order)
      if (arr[j] >= pivot) {
         i++; // Increment index of smaller element
         swap(&arr[i], &arr[j]);
```

```
        }
    }
    swap(&arr[i + 1], &arr[high]); // Place the pivot in its correct sorted position
    return (i + 1);
}

// Function to perform Quick Sort (descending order)
// arr[]: The array to be sorted
// low: Starting index of the sub-array
// high: Ending index of the sub-array
void quicksort(char arr[], int low, int high) {
    if (low < high) {
        // pi is partitioning index, arr[pi] is now at right place
        int pi = partition(arr, low, high);

        // Separately sort elements before partition and after partition
        quicksort(arr, low, pi - 1);
        quicksort(arr, pi + 1, high);
    }
}

int main() {
    int n;
    scanf("%d", &n);

    char characters[n];

    for (int i = 0; i < n; i++) {
        char input;
        scanf(" %c", &input);
        characters[i] = input;
    }

    quicksort(characters, 0, n - 1);

    for (int i = 0; i < n; i++) {
        printf("%c ", characters[i]);
    }

    return 0;
}
```

Status : Correct **Marks : 10/10**