# Rajalakshmi Engineering College

Name: krithika narasimhan
Email: 240701277@rajalakshmi.edu.in
Roll no: 240701277
Phone: 9677451731
Branch: REC
Department: I CSE FC
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## NeoColab_REC_CS23231_DATA STRUCTURES

## REC_DS using C_Week 6_CY_Updated

Attempt : 1
Total Mark : 30
Marks Obtained : 30

## Section 1 : Coding

1. Problem Statement

Sheela wants to distribute cookies to her children, but each child will only be happy if the cookie size meets or exceeds their individual greed factor. She has a limited number of cookies and wants to make as many children happy as possible. Priya decides to sort both the greed factors and cookie sizes using QuickSort to efficiently match cookies with children. Your task is to help Sheela determine the maximum number of children that can be made happy.

### Input Format

The first line of input consists of an integer n, representing the number of children.

The second line contains n space-separated integers, where each integer represents the greed factor of a child.

The third line contains an integer m, representing the number of cookies.

The fourth line contains m space-separated integers, where each integer represents the size of a cookie.

*Output Format*

The output prints a single integer, representing the maximum number of children that can be made happy.

Refer to the sample output for formatting specifications.

*Sample Test Case*

Input: 3
1 2 3
2
1 1
Output: The child with greed factor: 1

*Answer*

```java
import java.util.Scanner;

class Solution {

    private void quickSort(int[] nums, int low, int high) {
        if (low < high) {
            int pivot = partition(nums, low, high);
            quickSort(nums, low, pivot - 1);
            quickSort(nums, pivot + 1, high);
        }
    }

    private int partition(int[] nums, int low, int high) {
        int pivot = nums[high];
        int i = low - 1;

        for (int j = low; j < high; j++) {
            if (nums[j] <= pivot) {
                i++;
```

```java
            int temp = nums[i];
            nums[i] = nums[j];
            nums[j] = temp;
        }
    }

    int temp = nums[i + 1];
    nums[i + 1] = nums[high];
    nums[high] = temp;

    return i + 1;
}

public int findContentChildren(int[] g, int[] s) {
    if (s.length == 0) return 0;

    quickSort(g, 0, g.length - 1);
    quickSort(s, 0, s.length - 1);

    int maxNum = 0;
    int cookieIndex = s.length - 1;
    int childIndex = g.length - 1;
    while (cookieIndex >= 0 && childIndex >= 0) {
        if (s[cookieIndex] >= g[childIndex]) {
            maxNum++;
            cookieIndex--;
            childIndex--;
        } else {
            childIndex--;
        }
    }

    return maxNum;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    int numChildren = scanner.nextInt();
    int[] g = new int[numChildren];
    for (int i = 0; i < numChildren; i++) {
        g[i] = scanner.nextInt();
```

```
    }

    int numCookies = scanner.nextInt();
    int[] s = new int[numCookies];
    for (int i = 0; i < numCookies; i++) {
      s[i] = scanner.nextInt();
    }

    Solution solution = new Solution();
    int result = solution.findContentChildren(g, s);

    System.out.println("The child with greed factor: " + result);
  }
}
```

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


2.  Problem Statement

Reshma is passionate about sorting algorithms and has recently learned about the merge sort algorithm. She wants to implement a program that utilizes the merge sort algorithm to sort an array of integers, both positive and negative, in ascending order.

Help her in implementing the program.

*Input Format*

The first line of input consists of an integer N, representing the number of elements in the array.

The second line of input consists of N space-separated integers, representing the elements of the array.

*Output Format*

The output prints N space-separated integers, representing the array elements sorted in ascending order.

Refer to the sample output for formatting specifications.

*Sample Test Case*

Input: 9
5 -3 0 12 7 -8 2 1 6
Output: -8 -3 0 1 2 5 6 7 12

*Answer*

```c
#include <stdio.h>

void merge(int arr[], int left[], int right[], int left_len, int right_len) {
    int i = 0, j = 0, k = 0;

    while (i < left_len && j < right_len) {
        if (left[i] < right[j]) {
            arr[k] = left[i];
            i++;
        } else {
            arr[k] = right[j];
            j++;
        }
        k++;
    }

    while (i < left_len) {
        arr[k] = left[i];
        i++;
        k++;
    }

    while (j < right_len) {
        arr[k] = right[j];
        j++;
        k++;
    }
}

void mergeSort(int arr[], int n) {
    if (n <= 1) return;

    int mid = n / 2;
    int left[mid];
```

```c
    int right[n - mid];

    for (int i = 0; i < mid; i++) {
        left[i] = arr[i];
    }

    for (int i = mid; i < n; i++) {
        right[i - mid] = arr[i];
    }

    mergeSort(left, mid);
    mergeSort(right, n - mid);
    merge(arr, left, right, mid, n - mid);
}
int main() {
    int n;
    scanf("%d", &n);

    int arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    mergeSort(arr, n);

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

    return 0;
}
```

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

3.  Problem Statement

Meera is organizing her art supplies, which are represented as a list of integers: red (0), white (1), and blue (2). She needs to sort these supplies so that all items of the same color are adjacent, in the order red, white, and blue. To achieve this efficiently, Meera decides to use QuickSort to sort the

items. Can you help Meera arrange her supplies in the desired order?

### Input Format

The first line of input consists of an integer n, representing the number of items in the list.

The second line consists of n space-separated integers, where each integer is either 0 (red), 1 (white), or 2 (blue).

### Output Format

The output prints the sorted list of integers in a single line, where integers are arranged in the order red (0), white (1), and blue (2).

Refer to the sample output for formatting specifications.

### Sample Test Case

Input: 6
2 0 2 1 1 0
Output: Sorted colors:
0 0 1 1 2 2

### Answer

```java
import java.util.Scanner;

class Solution {

    private void quickSort(int[] nums, int low, int high) {
        if (low < high) {
            int pivotIndex = partition(nums, low, high);
            quickSort(nums, low, pivotIndex - 1);
            quickSort(nums, pivotIndex + 1, high);
        }
    }

    private int partition(int[] nums, int low, int high) {
        int pivot = nums[high];
        int i = low - 1;
```

```java
        for (int j = low; j < high; j++) {
            if (nums[j] <= pivot) {
                i++;
                int temp = nums[i];
                nums[i] = nums[j];
                nums[j] = temp;
            }
        }

        int temp = nums[i + 1];
        nums[i + 1] = nums[high];
        nums[high] = temp;

        return i + 1;
    }

    public void sortColors(int[] nums) {
        quickSort(nums, 0, nums.length - 1);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int n = scanner.nextInt();
        int[] nums = new int[n];

        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }

        Solution solution = new Solution();
        solution.sortColors(nums);

        System.out.println("Sorted colors:");
        for (int num : nums) {
            System.out.print(num + " ");
        }
    }
}
```

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