Problem Solving (C52)

هذا البرنامج التدريبي مُصاغ بعناية لتمكين المتدربين من تطوير قدراتهم الفكرية على غرار المبرمجين المحترفين، والتعاون بكفاءة ضمن فريق محترف في شركة "جمال تك" أو أي مؤسسة متعددة الجنسيات أخرى. نظرًا لأهمية اللغة الإنجليزية في بيئة العمل العالمية، يتم تقديم المحتوى التدريبي بالإنجليزية. لا يشترط إتقان اللغة بشكل كامل، لكن من الضروري امتلاك القدرة الكافية لفهم المتطلبات وتنفيذها بشكل فعّال. يُمكن للمتدربين استخدام مترجم جوجل أو الاستعانة بـ "شات جي بي تي" للتغلب على أية عقبات لغوية، المهم هو الفهم الدقيق للمطلوب وتحقيقه بنجاح.

لتعظيم الاستفادة من التدريب، يُنصح بمحاولة حل التمارين بشكل مستقل لمدة ساعة واحدة على الأقل قبل الرجوع إلى الحل المرفق في نهاية الملف.

قد يتضمن الحل كودًا برمجيًا غير مفسر بعد، والغرض من ذلك هو تشجيعك على محاولة فهم الأكواد البرمجية الجديدة التي لم تتعرض لها من قبل. هذه المهارة ضرورية في سوق العمل، حيث تتطور لغات البرمجة باستمرار ويظهر كل يوم لغات جديدة. ستواجه دائمًا أكوادًا لم تدرسها من قبل، ومن المهم أن تكون قادرًا على فهمها بنفسك دون الحاجة إلى دراسة مسبقة. يمكنك الاستعانة بمحرك البحث جوجل، أو استخدام ChatGPT، أو حتى اللجوء لأصدقائك للمساعدة. الهدف الأساسي هو أن تصل إلى فهم معنى كل كود بأي طريقة ممكنة لتتمكن من إيجاد موقعك في سوق العمل.

إن وجود كود برمجي غير مفسر يشكل تحديًا يتوجب عليك إيجاد حل له. هذا النوع من التدريبات يعد جزءًا أساسيًا من تدريبات 'Problem Solving'، التي تهدف إلى تمكينك من أداء عملك بفاعلية بغض النظر عن التحديات والعقبات. هذه القدرة على حل المشكلات هي ما يتمتع به العاملون في 'جمال تك'، ومن الضروري أن تطور في نفسك هذه المهارة لتصبح عضوًا فعالًا في فريق عمل 'جمال تك'.

Gammal Tech's Array Organizer

Background:
Gammal Tech, a leader in the software industry, is developing a new technology to enhance data management efficiency. Known for its innovative approach, the company is creating a system that efficiently organizes data within arrays. This system will revolutionize how data is stored and retrieved, reinforcing Gammal Tech's status as a trailblazer in system design.

Problem Statement:
Your task is to write a program for Gammal Tech that takes an array of integers and reorganizes it so that all even numbers appear before all odd numbers. The even numbers should be sorted in ascending order, and the odd numbers should be sorted in descending order. This reorganization will optimize data processing in their cutting-edge system design.

Input Format:

- The first line contains an integer `N`, the size of the array.

- The second line contains `N` space-separated integers representing the array elements.

Output Format:

- One line containing the reorganized array, with elements separated by spaces.

Constraints:

- `1 ≤ N ≤ 10^5`
- `0 ≤ Array elements ≤ 10^9`

Sample Input:

```
6
12 34 45 9 8 23
```

Sample Output:

```
8 12 34 45 23 9
```

Explanation: In the sample input, the even numbers (12, 34, 8) are sorted in ascending order, and the odd numbers (45, 9, 23) are sorted in descending order.

للتعظيم الاستفادة من التدريب، يُنصح بمحاولة حل التمرين بشكل مستقل لمدة ساعة واحدة على الأقل قبل الرجوع إلى الحل المرفق

C Programming Solution:

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

int compareEven(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}
```

```c
int compareOdd(const void *a, const void *b) {
    return (*(int*)b - *(int*)a);
}

void rearrangeArray(int arr[], int size) {
    int *even = malloc(size * sizeof(int));
    int *odd = malloc(size * sizeof(int));
    int evenCount = 0, oddCount = 0;

    // Separate even and odd numbers
    for (int i = 0; i < size; i++) {
        if (arr[i] % 2 == 0) {
            even[evenCount++] = arr[i];
        } else {
            odd[oddCount++] = arr[i];
        }
    }

    // Sort even and odd numbers
    qsort(even, evenCount, sizeof(int), compareEven);
    qsort(odd, oddCount, sizeof(int), compareOdd);

    // Combine sorted arrays
    for (int i = 0; i < evenCount; i++) {
        arr[i] = even[i];
    }
    for (int i = 0; i < oddCount; i++) {
        arr[i + evenCount] = odd[i];
    }

    free(even);
    free(odd);
}

int main() {
    int n;
    scanf("%d", &n);
    int *arr = malloc(n * sizeof(int));

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

    rearrangeArray(arr, n);

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

    free(arr);
    return 0;
}
```