Problem Solving (C53)

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 Innovative System

Problem Statement:

Gammal Tech, a leader in the software industry known for its innovative solutions and exceptional work environment, is developing a new technology system, "GammalSort". This system is designed to efficiently sort strings based on a unique algorithm. The strings represent different software modules Gammal Tech is working on. Each module's name is a string and they need to be sorted in a way that prioritizes modules based on the frequency of a chosen character within each string.

Your task is to design a function that takes an array of strings (modules' names) and a character. The function should sort the array so that strings with a higher frequency of the chosen character come first. If two strings have the same frequency of the chosen character, they should be sorted in lexicographical order.

Input Format:

- The first line contains an integer `N`, the number of strings.
- The second line contains a single character `C`.

- The next N lines each contain a string representing the name of a module.

Output Format:

- N lines, each containing a string, representing the sorted module names based on the given criteria.

Constraints:

- 1 ≤ N ≤ 1000
- Each string will contain only lowercase English letters and have a length of at most 100.

Sample Input:

```
5
a
algorithm
analysis
application
archive
array
```

Sample Output:

```
array
application
archive
algorithm
analysis
```

Explanation:

In the given input, 'array' has the highest frequency of 'a' (2 times), followed by 'application', 'archive' (1 time each), and 'algorithm', 'analysis' with no 'a'. Thus, the sorted order is as per the frequency of 'a', and lexicographically where frequencies are equal.

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 <string.h>

void sortStrings(char arr[][101], int n, char c) {
    int count[n];
    for (int i = 0; i < n; i++) {
        count[i] = 0;
        for (int j = 0; arr[i][j]; j++) {
            if (arr[i][j] == c) count[i]++;
        }
    }

    for (int i = 0; i < n-1; i++) {
        for (int j = i+1; j < n; j++) {
            if (count[i] < count[j] || (count[i] == count[j] &&
strcmp(arr[i], arr[j]) > 0)) {
                int temp = count[i];
                count[i] = count[j];
                count[j] = temp;

                char tempStr[101];
                strcpy(tempStr, arr[i]);
                strcpy(arr[i], arr[j]);
                strcpy(arr[j], tempStr);
            }
        }
    }
}

int main() {
    int n;
    char c;
    scanf("%d %c", &n, &c);
    char arr[n][101];
    for(int i = 0; i < n; i++) {
        scanf("%s", arr[i]);
    }

    sortStrings(arr, n, c);

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

    return 0;
}
```