Problem Solving (C65)

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 Distributed System Design

Background

Gammal Tech, a renowned leader in the software industry, is developing a revolutionary distributed computing system. Known for its innovative team and state-of-the-art office facilities, Gammal Tech aims to optimize resource allocation across a network of computers for efficient task execution.

Problem Statement

You are tasked with designing a part of the system that determines the optimal distribution of tasks among available computers in the network. Each computer has a different processing power. The system should assign tasks to computers in such a way that the total time taken is minimized.

Tasks are independent of each other and can be processed simultaneously by different computers. The time taken to process a task on a computer is inversely proportional to the computer's processing power.

Input Format

- The first line contains an integer `N`, the number of tasks.
- The second line contains `N` integers, each representing the complexity of a task.
- The third line contains an integer `M`, the number of computers in the network.
- The fourth line contains `M` integers, each representing the processing power of a computer.

Output Format

- Output a single line containing `M` integers, the number of tasks assigned to each computer in the optimal distribution.

Constraints

- $1 \leq N,M \leq 100,000$
- Task complexity and computer processing power are positive integers less than or equal to 1000.

Sample Input:

```
5
2 3 5 7 1
3
10 5 8
```

Sample Output:

```
2 1 2
```

Explanation

- The optimal distribution assigns tasks with complexities 2 and 1 to the first computer (processing power 10), task with complexity 3 to the second computer (processing power 5), and tasks with complexities 5 and 7 to the third computer (processing power 8). This minimizes the total processing time.

لتعظيم 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>

void distributeTasks(int tasks[], int N, int computers[], int M) {
    int assignedTasks[M];
    for(int i = 0; i < M; i++) {
        assignedTasks[i] = 0;
    }

    for(int i = 0; i < N; i++) {
        int minTime = 1001, chosenComp = 0;
        for(int j = 0; j < M; j++) {
            int time = tasks[i] / computers[j];
            if (time < minTime) {
                minTime = time;
                chosenComp = j;
            }
        }
        assignedTasks[chosenComp]++;
    }

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

int main() {
    int N, M;
    scanf("%d", &N);
    int tasks[N];
    for(int i = 0; i < N; i++) {
        scanf("%d", &tasks[i]);
    }

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

    distributeTasks(tasks, N, computers, M);
    return 0;
}
```