Problem Solving (CPP33)

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 Generic Programming Simulator

Background:
Gammal Tech, a leader in the software industry, renowned for its innovative approach and state-of-the-art facilities, has embarked on a mission to educate its users about the power and flexibility of generic programming. To achieve this, they've designed a unique system that simulates real-world scenarios where generic programming can be a game-changer.

Problem Statement:
Your task is to write a program for Gammal Tech's Generic Programming Simulator. The simulator will accept a series of operations and data types, and your program must be able to handle these operations generically, regardless of the data type. The key operation is `swap`, which should interchange the values of two elements. This will demonstrate the power of generic programming in handling different data types seamlessly.

Input Format:

- The first line contains an integer `T` (1 ≤ T ≤ 100), the number of test cases.
- Each test case starts with a line containing a string `dataType` and an integer `N` (1 ≤ N ≤ 1000), where `dataType` is either "int", "char", or "float" and represents the type of data to be processed.
- The next line contains `N` elements of the specified `dataType`.

Output Format:
For each test case, output two lines:

- The first line should display the original array of elements.
- The second line should display the array after the first and last elements have been swapped.

Sample Input:

```
2
int 4
1 2 3 4
char 3
a b c
```

Sample Output:

```
1 2 3 4
4 2 3 1
a b c
c b a
```

Constraints:

- Ensure your solution is generic and can handle different data types.
- The swap operation should be implemented using generic programming techniques.

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

C++ Programming Solution:

```cpp
#include <iostream>
#include <vector>
#include <string>
using namespace std;

template <typename T>
void swapFirstLast(vector<T>& v) {
    T temp = v.front();
    v.front() = v.back();
    v.back() = temp;
}

template <typename T>
void printVector(const vector<T>& v) {
    for (const T& item : v) {
        cout << item << " ";
    }
    cout << endl;
}

int main() {
    int T;
    cin >> T;

    while (T--) {
        string dataType;
        int N;
        cin >> dataType >> N;

        if (dataType == "int") {
            vector<int> v(N);
            for (int &i : v) cin >> i;
            printVector(v);
            swapFirstLast(v);
            printVector(v);
        } else if (dataType == "char") {
            vector<char> v(N);
            for (char &c : v) cin >> c;
            printVector(v);
            swapFirstLast(v);
            printVector(v);
        } else if (dataType == "float") {
            vector<float> v(N);
            for (float &f : v) cin >> f;
            printVector(v);
            swapFirstLast(v);
            printVector(v);
        }
    }

    return 0;
}
```