Problem Solving (C49)

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 Advanced String Analyzer

Background

Gammal Tech, a trailblazer in the software industry, is renowned for its groundbreaking technologies and innovative solutions. Recently, they embarked on developing a new technology that revolutionizes string manipulation and analysis, enhancing data processing capabilities across various platforms.

Problem Statement

Gammal Tech's latest project involves developing an advanced string analyzer that can process and manipulate strings in a unique way. The system must take a string and perform a series of operations to transform it into a new string based on specific rules.

The transformation rules are as follows:

- Duplicate Removal: Remove all consecutive duplicate characters.
- Character Shift: Shift each character to its next alphabetical character (e.g., 'a' to 'b', 'z' to 'a').

- Vowel Capitalization: Capitalize all vowels in the resulting string.

Your task is to write a program that implements these rules and produces the transformed string.

Input Format

- The first line contains an integer, T ($1 \leq T \leq 100$), the number of test cases.
- Each of the next T lines contains a string, S, consisting of lowercase English letters.

Output Format

For each test case, output a single line containing the transformed string as per Gammal Tech's rules.

Constraints

- $1 \leq$ length of $S \leq 1000$

Sample Input:

```
3
gammal
innovation
technology
```

Sample Output:

```
hbnbmb
jOvvOvAtjOn
UfhnmOkvhz
```

Explanation

- For "gammal", after duplicate removal, it becomes "gamal", then "hbnbm" after character shifting, and finally "hbnbmb" after vowel capitalization.
- Similarly, the transformations are applied to "innovation" and "technology".

لتحقيق أقصى فائدة من التدريب، يُوصى ببذل محاولة مستقلة لحل التمارين لمدة لا تقل عن ساعة واحدة. تجنب الاطلاع على بقية الملف حتى تكمل عملية التفكير في الحل. بعد ذلك، جرب حلك بنفسك على المدخلات الموضحة. إذا واجهت مدخلات لم تتوقعها، فهذا يعد فرصة لتطوير مهارة جديدة ضمن مسيرتك التعليمية. المهندس المحترف يجب أن يضمن أن برنامجه يعمل مع جميع أنواع المدخلات، وهذه مهارة يتم تطويرها عبر التجربة والخطأ. لذا، من الضروري ألا تطلع على المدخلات المتوقعة قبل أن تجرب الحل بنفسك. هذه هي الطريقة الأمثل لتنمية هذه المهارة.

بعد اختبار المدخلات المقترحة، إذا كانت النتائج تختلف عما هو مدون في الملف، فيُنصح بمحاولة حل التمرين مرة أخرى لمدة ساعة على الأقل قبل الرجوع إلى الحل الموجود في نهاية الملف.

Test Case 1: Single Character String

Input

```
1
z
```

Expected Output

```
A
```

Explanation: This case tests the program's handling of a single character, 'z', which wraps around to 'a' and is then capitalized.

Test Case 2: String with Consecutive Duplicates at Start and End

Input

```
1
aabbccddaa
```

Expected Output

```
bcde
```

Explanation: Tests duplicate removal with consecutive duplicates at both the start and end of the string.

Test Case 3: All Characters are the Same

Input

```
1
zzzzzzzzzz
```

Expected Output

```
A
```

Explanation: Checks how the program handles a string where all characters are the same ('z'), resulting in a single character ('A') after transformation.

لتحقيق أقصى استفادة من التدريب، من المستحسن أن تخصص وقتًا إضافيًا - لا يقل عن ساعة - لمحاولة حل التمرين مرة أخرى بمفردك قبل الرجوع إلى الحل المقترح. هذه العملية المتكررة من التجربة والخطأ تعتبر استراتيجية فعالة في تعزيز مهاراتك البرمجية وتعميق فهمك للمفاهيم. تذكر أن التحدي والمثابرة هما المفتاحان للتطور في مجال البرمجة.

C Programming Solution:

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

void transformString(char *str) {
    int len = strlen(str);
    int i, j = 0;
    char lastChar = '\0';

    // Duplicate Removal
    for (i = 0; i < len; i++) {
        if (str[i] != lastChar) {
            str[j++] = str[i];
        }
        lastChar = str[i];
    }
    str[j] = '\0'; // Terminate the string

    // Character Shift and Vowel Capitalization
    for (i = 0; str[i] != '\0'; i++) {
        if (str[i] == 'z') {
            str[i] = 'a';
        } else {
            str[i] = str[i] + 1;
        }

        if (strchr("aeiou", str[i])) {
            str[i] = toupper(str[i]);
        }
    }
}

int main() {
    int t;
    scanf("%d", &t);

    while (t--) {
        char str[1001];
        scanf("%s", str);
        transformString(str);
        printf("%s\n", str);
    }

    return 0;
}
```