

英语课的单词接龙 ☆☆

题目描述

英语老师为了活跃课堂气氛，组织大家玩单词接龙游戏。游戏规则如下：后一个单词的首字母必须与前一个单词的尾字母相同（不区分大小写，例如 Dog 后可以接 Girl）。

现老师记录了班长依次说出的 N 个单词，请你编写程序判断该接龙序列是否合法。

输入格式

第一行一个整数 N 。

接下来 N 行，每行一个由英文字母组成的字符串。

输出格式

如果全部符合规则，输出 **Valid**。

否则输出一个整数 **X**，表示第 X 个单词出错了（从 1 开始计数）。

输入输出样例

输入 #1	输出 #1
4 Apple Egg Good Day	Valid

算法分析

本题主要考察 **字符串的基本操作** 与 **字符大小写转换**。

1. 数据存储

由于题目需要输出出错单词的编号 X ，我们可以使用 **string** 数组（或 **vector<string>**）来存储所有单词，下标建议从 1 开始，方便直接对应题目要求的编号。

2. 核心逻辑 ($O(N)$)

我们需要从第 2 个单词开始遍历到第 N 个单词。

对于每一个单词 $s[i]$ ，检查首字母 $s[i][0]$ 是否与前一个单词 $s[i - 1]$ 的尾字母匹配。

- 获取前一个单词的尾字母：可以使用 $s[i-1].back()$ 或者 $s[i-1][s[i-1].size()-1]$ 。

- 忽略大小写：这是本题的易错点。`'A'` 和 `'a'` 在 ASCII 码中是不相等的。比较时需要统一转换为大写或小写。

3. 字符处理：忽略大小写

题目要求不区分大小写，例如 'A' 和 'a' 应视为相同。我们可以将首尾字母统一转换为小写（或大写）后再进行比较。这里介绍两种常用方法：

- **方法一：调用标准库函数（推荐）**

C++ 的 `<cctype>` 头文件提供了 `tolower(c)` 函数，可以将字符转换为小写（如果是符号或已经是小写，则保持不变）。

```
#include <cctype>
// ...
char head = tolower(s[i][0]);           // 当前单词首字母转小写
char tail = tolower(s[i-1].back());     // 前一单词尾字母转小写

if (head != tail) {
    cout << i << endl; // 发现不匹配，输出编号
    return 0;           // 直接结束程序
}
```

- **方法二：利用 ASCII 码手动转换（底层原理）**

在 ASCII 码表中，大写字母 `A` (65) 与小写字母 `a` (97) 之间相差 **32**。因此，判断一个字符如果在大写范围内 (`A ~ Z`)，只要加上 32 即可变为对应的小写字母。

```
char head = s[i][0];
char tail = s[i-1].back();

// 手动转小写：如果是大写字母，加 32 变小写
if (head >= 'A' && head <= 'Z') head += 32;
if (tail >= 'A' && tail <= 'Z') tail += 32;

if (head != tail) {
    cout << i << endl;
    return 0;
}
```

4. 完整流程

如果循环结束后都没有触发 `return 0`，说明所有单词都符合规则，最后输出 `Valid`。

拓展思考

进阶变式：最长单词接龙 (DFS/图论)

本题只是验证给定的序列是否合法。在竞赛（如 NOIP 2000 提高组）中，更经典的问题是：

给出一堆单词，如果不规定顺序，问你能拼接出的**最长接龙序列**有多长？

思路：这就从“模拟”变成了“搜索”。我们需要把每个单词看作图中的一个节点，如果单词 A 的尾部能接单词 B 的首部，就建立一条有向边（或在 DFS 中递归跳转）。由于可能存在环或者单词复用次数限制，通常使用 **深度优先搜索 (DFS)** 来寻找最长路径。