预处理器（preprocessor）

【题目描述】

宏是 C/C++ 语言的一项特性，它根据预先定义的规则进行文本替换（也被称为“宏展开”），能够实现定义常量、简化代码重复输入等功能。例如：

```
#define PI 3.14159
double area = PI * r * r;
```

以上代码经过宏展开后变为：

```
double area = 3.14159 * r * r;
```

其中，宏定义命令变成了空行，而其他行中的宏被展开成了规则定义的文本。

C/C++ 语言代码在编译时对宏的处理由**预处理器**完成。你的任务是实现一个简化版的预处理器，要求如下：

- 代码由**行**组成，每行除行末的换行符外，均由可打印 ASCII 字符（ASCII 码范围 32—126）组成。每行要么是**预处理命令**（以 # 开头），要么是**普通文本**（其他情况）。
- 预处理器逐行处理代码，
  - 如果是预处理命令，执行该命令，并输出一个空行。
  - 如果是普通文本，对其进行宏展开并输出结果。
- 预处理命令有两种，分别是宏定义命令 `#define` 和取消宏定义命令 `#undef`。
  - 宏定义命令的格式为 `#define <name> <content>`,其中第一部分 `#define` 是命令名，第二部分 `<name>` 是要定义的宏的名字，第三部分 `<content>` 是要定义的宏的展开内容。
  - 取消宏定义命令的格式为 `#undef <name>`，其中第一部分 `#undef` 是命令名，第二部分 `<name>` 是要取消的宏的名字。

  以上两种预处理命令中，相邻两部分之间都严格用一个空格分隔。`<name>` 是由大小写字母和数字以及下划线组成的**标识符**（一个或多个字符），`<content>` 可以包含任意可打印 ASCII 字符（零个或多个字符）。一个宏定义的**有效范围**是从它定义所在行开始到后续最近的宏名匹配的取消定义所在行为止（如果没有对应的取消定义，则有效范围一直覆盖到文件结束）。

对普通文本进行宏展开时，将一行文本中每段**连续极长**的大小写字母和数字以及下划线视为标识符（而不是其中一部分），其余为**其他字符**。从左到右依次对文本中的标识符进行宏展开：

1. 如果该标识符是有效的宏名，则用对应的展开内容替换它，此时该宏名进入正在展开的状态，直到本流程结束；否则原样保留宏名。例如，若宏 `A` 定义为 `b`，则

文本 A 展开结果为 b（发生替换），文本 B 展开结果仍然为 B（未定义，不替换），文本 AA 展开结果仍然为 AA（AA 是不同于 A 的另一个标识符，未定义），而文本 A*B 开展结果为 b*B。

2. 替换发生后，如果展开内容中包含标识符，重复应用以上的展开操作，称为“多次展开”。例如，若宏 A 定义为 B，宏 B 定义为 c，则文本 A 的展开结果为 c。
3. 如果待展开的宏名与正在进行展开的某个宏名相同，称为“递归展开”，此时该宏名不再展开。本规则用来防止无限递归展开。例如，若宏 A 定义为 B+a，宏 B 定义为 A+b，则文本 A 展开结果为 A+b+a，由于最初的 A 处于正在展开状态，因此 A+b+a 里的 A 不再展开。
4. 其他字符原样保留。

注意：出于简化的目的，本题的要求与 C/C++ 语言标准里的描述不完全一致，请以上面的要求为准。最明显的区别是本题只有标识符和其他字符两类词法单元，没有数值、字符串、注释等。

【输入格式】

从文件 ***preprocessor.in*** 中读入数据。

输入的第一行包含一个正整数 n，表示要处理的代码行数。

接下来的 n 行是要处理的代码。

【输出格式】

输出到文件 ***preprocessor.out*** 中。

输出 n 行，为输入逐行预处理后的结果。

【样例 1 输入】

```
5
#define BEGIN {
#define END }
#define INTEGER int
class C BEGIN INTEGER x; END;
INTEGER main() BEGIN C c; END
```

【样例 1 输出】

```

```

```
class C { int x; };
int main() { C c; }
```

【样例 2】

见选手目录下的 ***preprocessor/preprocessor2.in*** 与 ***preprocessor/preprocessor2.ans***。

【样例 3】

见选手目录下的 ***preprocessor/preprocessor3.in*** 与 ***preprocessor/preprocessor3.ans***。

【数据范围】

对 20% 的数据，不会出现宏定义命令 `#define` 和宏取消定义命令 `#undef`。

对另外 20% 的数据，不会出现多次展开的情况，且不会出现宏取消定义命令 `#undef`。

对另外 20% 的数据，不会出现多次展开的情况。

对另外 20% 的数据，不会出现递归展开的情况。

对其余数据，无特殊限制。

对 100% 的数据，$n \leq 100$，输入的每行字符数都不超过 100，且保证输出的每行字符数都不超过 1 000（字符数均不计行末换行符）。保证输入数据中的预处理命令都是合法的，包含但不限于：

- `#` 字符只会出现在预处理命令所在行的第一个字符的位置，其他任何位置（包括预处理命令和普通文本）都不会出现 `#` 字符。
- 宏定义和取消定义命令的格式是正确的，严格遵循题面所描述的格式。
- 同一个宏在取消定义之前不会被再次定义。
- 要取消定义的宏在之前被定义过且还没有被取消过。

也就是说，**你不需要做任何语法和语义的错误检查**。

【提示】

本题进行输入时建议使用 C++ 语言的按行读入字符串功能，示例如下：

```
#include <iostream>
#include <string>
using namespace std;
string line;
// 从 cin 读入一行，放入 line 中（换行符被舍弃）
```

```
getline(cin, line);
```

也可以使用 C 语言提供的 fgets 函数，示例如下：

```
#include <stdio.h>
#define MAX_LEN 200
char line[MAX_LEN];
// 从 stdin 读入一行，放入 line 中（包含换行符）
fgets(line, MAX_LEN, stdin);
```

注意：在读取行数 n 之后可能需要额外读取一行以忽略其后的换行符。