題目 D
巨集危機

執行時間限制: 5 秒

相信大家都知道 C/C++ 語言有提供巨集(macro)的功能，我們可以藉由 `#define` 來實作之，寫出各式各樣五花八門的巨集。若不知道也沒有關係，我們可以參考以下的程式碼：

```
#define SQUARE(x) ((x) * (x))
#define CUBE(x)   ((x) * (x) * (x))
```

有了這兩行巨集宣告，我們就可以在往後的程式碼中書寫如以下的述句，就如同我們宣告了兩個名叫 `SQUARE` 和 `CUBE` 的函式(function)一樣：

- 對於 C 語言：

```
scanf("%d", &x);
printf("%d %d\n", SQUARE(x), CUBE(x));
```

- 對於 C++ 語言：

```
cin >> x;
cout << SQUARE(x) << " " << CUBE(x) << "\n";
```

其概念爲，在編譯時我們便會將所書寫的巨集代入程式碼中，你可以當作是一種置換(replace)的動作，置換之後的程式碼大概會長的像下面這樣：

- 對於 C 語言：

```
scanf("%d", &x);
printf("%d %d\n", ((x) * (x)), ((x) * (x) * (x)));
```

- 對於 C++ 語言：

```
cin >> x;
cout << ((x) * (x)) << " " << ((x) * (x) * (x)) << "\n";
```

然而，書寫巨集需要非常小心，一旦有一點疏忽，很容易就會造成錯誤的結果。舉例來說，如果我們因爲一時偷懶，將以上的程式碼寫成：

```
#define SQUARE(x) (x * x)
```

則我們將會得到錯誤的結果。爲什麼呢？因爲假設我們的 x 爲 5 + 7，則我們會認爲 SQUARE(5 + 7) 的結果應該爲 $(5+7)^2 = 144$；可是，如果我們因爲一時的疏失而少寫了一些括號，則結果將會是出乎意料的(如果你不知道原因的話)：你會得到 47。原因在於，我們在展開巨集的過程中，編譯器所做的動作是「直接代換 x 的值」。就剛剛上面錯誤的例子會是：

- 對於 C 語言：

```
printf("%d\n", SQUARE(5 + 7));
 => printf("%d\n", (5 + 7 * 5 + 7));
```

- 對於 C++ 語言：

```
cout << SQUARE(5 + 7) << "\n";
 => cout << (5 + 7 * 5 + 7) << "\n";
```

發現到什麼問題了嗎？

爲了避免這種錯誤的發生，有些人會開始瘋狂的加括號，每一步都加括號！當然，這樣並沒有不好，某方面來說這樣的編程習慣或許還是比較好的。但是，達克皮(Darkpi)很不喜歡有一堆括號的程式碼，他覺得只要答案對就好了何必要那麼多的括號？ 因此，他想要問說，對於一個巨集運算式，在合理(不影響正確性)的情況下，最多可以拿掉幾對括號？

我們保證對於所有輸入的巨集其名稱([MACRO_NAME])都只包含英文大寫字母和底線 _ ，並且長度不超過 50。而對於所有輸入的巨集其運算式([OPERATION])，都只包含 變數 x 、左括號 (、右括號) 、加法二元運算子 + 和乘法二元運算子 * ，並且長度不超過 40。爲了進一步的簡化題目，我們假設輸入的 x 都是一個只包含加(+)和乘(*)的整數運算式(例如：2 + 5 * 3 + 1)。意思是你只要保證對於所有 + 和 * 運算子組合出來的算式都能得出**正確的**答案，那麼就是一組合法的方案。所謂的「正確的」定義爲：假設你把 [OPERATION] 中的幾組括號拿掉了，得到一個新的式子 [NEW_OP]，則下面兩個函式所回傳的結果在無溢位狀態發生時必須是一樣的。

```
int [MACRO_NAME](int x) { return [NEW_OP]; }
int [MACRO_NAME](int x) { return [OPERATION]; }
```

■ 輸入檔說明

輸入的第一行有一個正整數 T，代表測試資料的組數 $(1 \leq T \leq 100)$。

之後緊接著有 T 行(因此整個輸入檔案將有 $T + 1$ 行)，每一行都恰好有一個已經寫好的巨集。

我們保證原本給定的巨集都是「安全的」，只是可能有多餘的不必要之括號。 輸入的巨集程式碼格式如下:

```
#define [MACRO_NAME](x) [OPERATION]
```

如果在 [OPERATION] 中有出現 + 或 * 等運算子，則其前後必定會有且僅有**各一個**空白。

上面說所說的「安全的」是指，所給定的巨集會跟下面的函式有相同的效果。(在題目對 x 的條件下)

```
int [MACRO_NAME](int x) { return [OPERATION]; }
```

■ 輸出檔說明

對於每一個輸入的巨集(共 T 組(行))，請輸出一個整數代表最多可以拿掉幾對括號而仍保持答案的正確性。

重要的是，我們規定無論你是怎麼消括號的，**最外層**的括號**一定要保留**，不可以刪除。

■ 範例輸入

```
4
#define SQUARE(x) (((x) * ((x))))
#define PLUS_TWO(x) ((x) + (x))
#define ADD_THREE_TIMES(x) (x + (x + x))
#define CANNOT_REMOVE(x) ((x) * (x + x))
```

■ 範例輸出

```
2
2
1
0
```