題目 H
古可魚語

執行時間限制: 30 秒

可魚國的考古學家們日前發現一處古代遺跡，裡面有以古可魚語書寫的眾多文物，經過語言學家老蚯的努力後，終於破譯出古可魚語的規則。但是文物實在太多了，不知何時才能將所有文物翻完，因此聰明的你想要寫個程式來幫幫他們！

古可魚語中每句話均由一對括號包起來，一句話包含若干個詞，詞跟詞之間會以至少一個空白或換行隔開。空白和換行可能會交錯出現，不過都同樣視為隔開以及排版用，沒有任何意義。

一句話形如 (**w_1 w_2 $\cdots$ w_n**)，其中 w_1 決定了這句話的句型。每個詞可以是一句話、一個數字或一個識別字。例如 **display** 是一個識別字，514 是一個數字，(**display** 514) 則是一句話。

保證數字只由 0123456789 組成，範圍在 0 到 2^{30} 之間，且不會有前導 0；識別字的開頭必然不在 0123456789 之內，且只會由 ASCII 碼 33 至 39、42 至 126 組成（ASCII 40 跟 41 是括號喲）。

而一個數字翻譯之後 …… 毫無反應，還是同一個數字；一個識別字翻譯後之結果視當下的情境而定（見如下 **define** 句型），而一句話翻譯之結果則視句型而定。

古可魚語中有下列幾種句型：

1. (**display** w)
   印出 w 翻譯之後的結果，保證 w 經過翻譯後是個數字，整句話翻譯後為 0。例如

   ```
   (display 514)
   (display (display 514))
   ```

   將會印出

   ```
   514
   514
   0
   ```

2. **(begin** w_1 w_2 $\cdots$ w_n**)**
依序翻譯 w_1 到 w_n，整句話翻譯後為 w_n 翻譯後的結果。例如

```
(display (begin
    (display 1)
    (display 2)
    3))
```

將會印出

```
1
2
3
```

3. **(if** w_1 w_2 w_3**)**
如果 w_1 翻譯後的結果不為 `0`，則整句話翻譯後為 w_2 翻譯後的結果，且 w_3 不予翻譯；如果 w_1 翻譯後為 `0`，則整句話翻譯後為 w_3 翻譯後的結果，且 w_2 不予翻譯。保證 w_1 翻譯後的結果為數字。例如

```
(if (display 514)
    (display 514514)
    (display 514514514))
```

將會印出

```
514
514514514
```

4. **(define** w_1 w_2**)**
先將 w_2 翻譯為 v_2，之後在當前情境中將 w_1 定義為 v_2，保證 w_1 是個識別字且不在 **display**、**begin**、**define**、**lambda**、+、-、< 中，重複 **define** 將會覆蓋之前的定義。整句話翻譯後亦為 w_2 翻譯後的結果。

例如

```
(if (display (define x 514))
    (display 514514)
    (display x))
(display x)
(define x 50216)
(display x)
```

將會印出

```
514
514
514
50216
```

5. **(+** w_1 w_2**)**
依序翻譯 w_1、w_2 為 v_1、v_2，則整句話翻譯後為 $v_1 + v_2$，保證 v_1、v_2 為數字。

(- w_1 w_2**)**
依序翻譯 w_1、w_2 為 v_1、v_2，則整句話翻譯後為 $v_1 - v_2$，保證 v_1、v_2 為數字。

(< w_1 w_2**)**
依序翻譯 w_1、w_2 為 v_1、v_2，如果 $v_1 < v_2$ 則整句話翻譯後為 1，否則為 0，保證 v_1、v_2 為數字。

例如

```
(if (< 514 50216)
    (display (+ 514 50216))
    (display (- 514 50216)))
```

將會印出

```
50730
```

6. **(lambda** (w_1 w_2 $\cdots$ w_n) b**)**
保證其中 w_1 到 w_n 均為識別字且兩兩相異。

注意！這是困惑眾人最久的句型。整句話翻譯後為一個「句型」搭配當下的「情境」，與程式設計中的函數非常類似，若 f 為本句翻譯後的結果，則**(**f p_1 p_2 $\cdots$ p_n**)** 將依下述規則翻譯：

(a) 依序翻譯 p_1 到 p_n，令結果為 v_1 到 v_n。

(b) 令 b 中的情境為 E_1，其中定義 w_1 到 w_n 為 v_1 到 v_n。

(c) 令翻譯 **(lambda** (w_1 w_2 $\cdots$ w_n) b**)** 時的情境為 E_2。

(d) 令翻譯 **(**f p_1 p_2 $\cdots$ p_n**)** 時的情境為 E_3。

(e) 翻譯 b，若在過程中遇到 **define**，則此 **define** 只作用在 E_1 中；若在過程中遇到識別字需要翻譯，則先嘗試由 E_1 中解讀，若失敗則再嘗試由 E_2 中解讀，若再失敗則嘗試由 E_3 中解讀。

例如

```
(define add1 (lambda (x) (+ x 1)))
(display (add1 2))
```

將會印出 3，而

```
(define y 1)
(define magic!! (lambda (x) (+ x (+ y z))))
(define x 100)
(define y 200)
(define z 400)
(display (magic!! 2))
```

將會印出 403。

■ 輸入說明

輸入檔中僅有一組測試資料，為以古可魚語寫成的若干句話。保證每句話均符合古可魚語之文法。

每句話可能涵蓋一行或多行，但同一行中只會有一句話。如果遇到以 "**;**" (不含雙引號) 開頭的行，那是考古學家的註記，請不要翻譯。註記不會夾雜在古可魚語中。

保證所有數字在運算過程中絕對值不超過 2^{30}。過程中翻譯詞句的總次數不會超過 10^7 個。

■ 輸出說明

印出所有翻譯過程中會印出的東西，一個一行。遇到考古學家的註記的時候依原樣直接輸出即可。

■ 範例輸入

```
; 3
(display (+ 1 2))

(define a 3)
; 7
(display (+ a 4))

(define add1 (lambda (x) (+ x 1)))
; 6
(display (add1 5))

(define pair (lambda (x y) (lambda (m) (m x y))))
(define first (lambda (x) (x (lambda (a b) a))))
(define second (lambda (x) (x (lambda (a b) b))))

(define a (pair 1 2))
; 1
(display (first a))
; 2
(display (second a))

(define [] (pair 1 1))
(define empty? first)
(define : (lambda (hd tl) (pair 0 (pair hd tl))))
(define head (lambda (xs) (first (second xs))))
(define tail (lambda (xs) (second (second xs))))

(define ++
  (lambda (xs ys)
    (if (empty? xs)
      ys
      (: (head xs) (++ (tail xs) ys)))))
```

```
33
34 (define map
35   (lambda (func xs)
36     (if (empty? xs)
37       []
38       (: (func (head xs)) (map func (tail xs))))))
39
40 (define display-list (lambda (xs) (map display xs)))
41
42 (define peter (: 5 (: 0 (: 2 (: 1 (: 6 []))))))
43 ; 5 0 2 1 6
44 (display-list peter)
45
46 (define filter
47   (lambda (pred xs)
48     (if (empty? xs)
49       []
50       (begin
51         (define xs' (filter pred (tail xs)))
52         (if (pred (head xs))
53           (: (head xs) xs')
54           xs')))))
55
56 (define ! (lambda (x) (if x 0 1)))
57 (define <= (lambda (x y) (! (< y x))))
58 (define > (lambda (x y) (< y x)))
59 (define sort
60   (lambda (xs)
61     (if (empty? xs)
62       []
63       (begin
64         (define x (head xs))
65         (define xs' (tail xs))
66         (define <=x (filter (lambda (y) (<= y x)) xs'))
67         (define  >x (filter (lambda (y) ( > y x)) xs'))
68         (++ (sort <=x) (: x (sort >x)))))))
69
70 ; 0 1 2 5 6
71 (display-list (sort peter))
72
```

```
(define take
  (lambda (xs)
    (lambda (n)
      (if n
        (: (head xs) ((take (tail xs)) (- n 1)))
        [])))))

; 5 0 2
(display-list ((take peter) 3))
(define take-peter (take peter))
; 5 0 2
(display-list (take-peter 3))

(define yin-yang
  (lambda ()
    (lambda (m)
      (m 0 (pair 0 (lambda (m)
                     (m 0 (pair 1 (yin-yang)))))))))

; 0 1 0 1 0
(display-list ((take (yin-yang)) 5))
```

■ 範例輸出

```
; 3
3
; 7
7
; 6
6
; 1
1
; 2
2
; 5 0 2 1 6
5
0
2
1
6
; 0 1 2 5 6
0
1
2
5
6
; 5 0 2
5
0
2
; 5 0 2
5
0
2
; 0 1 0 1 0
0
1
0
1
0
```