# 2013 網際網路程式設計全國大賽
# 高中組決賽

- 題目：本次比賽共 八 題 (含本封面共 28 頁)。
- 題目輸入：全部題目的輸入都來自**標準輸入**。
- 題目輸出：全部題目的輸出皆輸出到螢幕 (**標準輸出**)。
  輸出和裁判的答案必須完全一致，英文大小寫不同或多餘空白換行字元皆視為錯誤答案。
- 時間限制：每一題的執行時間限制如下表所示。
  其間執行的電腦上不會有別的動作、也不會使用鍵盤或滑鼠。
- 比賽中上傳之程式碼請依照以下規則命名：

  1. 若使用 C 做為比賽語言則命名為 `pa.c`, `pb.c`, 以此類推。
  2. 若使用 C++ 做為比賽語言則命名為 `pa.cpp`, `pb.cpp`，以此類推。

  未按照此規則命名之程式碼將可能因此得到 Compilation Error。
- `long long` 型別的整數使用方式請參考下一頁。
- `cin` 輸入經測試發現速度遠慢於 `scanf` 輸入，答題者若使用需自行承擔因輸入速度過慢導致 Time Limit Exceeded 的風險。

表 1: 題目資訊

| | 題目名稱 | 執行時間限制 |
|---|---|---|
| 題目 **A** | 可魚果運輸問題 | 10 秒 |
| 題目 **B** | 薑餅人的新樂園 | 1 秒 |
| 題目 **C** | 棒球練習場 | 1 秒 |
| 題目 **D** | 樣式圖像辨識 | 5 秒 |
| 題目 **E** | 可魚果贈送問題 | 5 秒 |
| 題目 **F** | 可魚果滾動問題 | 1 秒 |
| 題目 **G** | 胖胖天大大薯 II | 5 秒 |
| 題目 **H** | 古可魚語 | 30 秒 |

# 2013 網際網路程式設計全國大賽
# 解題程式輸入輸出範例

C 程式範例：

```c
#include <stdio.h>
int main(void)
{
    int cases, i;
    long long a, b;
    scanf("%d", &cases);
    for (i = 0; i < cases; i++)
    {
        scanf("%I64d %I64d", &a, &b);
        printf("%I64d\n", a + b);
    }
    return 0;
}
```

C++ 程式範例：

```cpp
#include <iostream>
int main()
{
    int cases;
    std::cin >> cases;
    for (int i = 0; i < cases; ++i)
    {
        long long a, b;
        std::cin >> a >> b;
        std::cout << a + b << std::endl;
    }
    return 0;
}
```

# 題目 A
# 可魚果運輸問題

**執行時間限制: 10 秒**

你是一個販賣可魚果的廠商。最近有一隻富有的大可魚花了他五百一十四分之一百四十五的財產訂購了大量的可魚果，導致你原本的物流機制出了些問題。無計可施的情況下，你只得把運輸可魚果的工作交給可魚國內有名的墨魚運輸公司處理。

然而，正如墨水一般黑，墨魚運輸公司雖然運輸量大、使命必達，但墨魚運輸公司所開的價碼異常昂貴。俗話說，賠錢的生意沒人作，作為一個商人，你為了避免利潤受到過多的壓縮，決定參與規劃運輸路線，使得運輸成本降到最低。

墨魚運輸公司提供了許多運輸方案，每個方案 $P_i$ 會從一個固定的起始城市 $A_i$ 運送東西到另一個固定的終點城市 $B_i$，每運輸一箱的可魚果，你就必須付給墨魚運輸公司 $C_i$ 可魚幣。另外，若兩個方案的終點與起點相接，則可以不花任何額外費用的將貨物轉過去。不過由於你的運輸量太大了，墨魚運輸公司決定祭出優惠，若用方案 $P_i$ 運輸了超過 $D_i$ 箱的可魚果，多出來的部份每箱改收優惠價 $C'_i$ 可魚幣。

你原本是隻數學挺好的可魚，但不幸的，你前些日子去東北方的日升之國談生意時感冒了，無法進行複雜的運算。你好奇運輸費究竟可以壓到多低呢？

## ■ 輸入檔說明

輸入的第一行有一個正整數 $T$ ($T \leq 20$)，代表測試資料的組數。

每筆測資的第一行有五個正整數 $N$ ($N \leq 100$), $M$ ($M \leq N(N-1)$), $S$, $E$ ($1 \leq S, E \leq N$), $F$ ($F \leq 10^9$)，各一個空白隔開。$N$ 代表可魚國內有幾個城市，$M$ 代表墨魚運輸公司提供幾種運輸方案。$S, E$ 分別代表你的工廠所在的城市以及你所要送去的城市。$F$ 代表那隻富有的大可魚訂購了多少箱的可魚果。

接下來 $M$ 行，每行有五個正整數 $A_i$, $B_i$ ($1 \leq A_i, B_i \leq N$), $C_i$, $D_i$ ($D_i \leq 10^9$), $C'_i$ ($C'_i \leq C_i \leq 50216$)，意思如同敍述中所說。

## ■ 輸出檔說明

對每筆測試資料請輸出一行，包含一個整數代表至少需要付給墨魚運輸公司多少可魚幣。

## ■ 註腳

其實你原本是隻滿耐寒的可魚，但你在一個僅有 2°C 的雨天中只穿了一件短袖棉 T 與外套，就感冒了。並且因為感冒，思路不清，你在日升之國的生意就給死對頭 0perasan 搶走了。

## ■ 範例輸入

```
3
4 4 1 4 1
1 2 1 1 1
2 4 5 1 3
1 3 1 1 1
3 4 6 1 1
4 4 1 4 2
1 2 1 1 1
2 4 5 1 3
1 3 1 1 1
3 4 6 1 1
2 1 1 2 999999999
1 2 50216 1000 50216
```

## ■ 範例輸出

```
6
9
50215999949784
```

# 題目 B
# 薑餅人的新樂園

**執行時間限制: 1 秒**

架設好全新的基地台後，薑餅人們的生活變得更加方便，部落也漸漸地恢復往日的蓬勃氣象。有一天，薑餅人工程師小薑在維修基地台時收到了來自地底下的不明訊號。小薑分析出訊號出現的位置之後便帶著鏟子與水壺前往地底探險，在那裡小薑發現了一處巨大的巧克力熔岩池，身為一個熱愛巧克力的薑餅人，他開心地吃了個痛快並且把水壺裝滿才回家。

從探險回來的那天起，小薑便經常收到來自地底下不同地方的訊號，而小薑也經常在實地探訪的過程中發現各種大量的點心。小薑希望讓部落的其他人也能夠共享這些點心，因此他向薑餅長老提出了一項建設計畫，打造地下的點心樂園。在計畫中總共有 $N$ 個有點心的地方分散在地下各處，為了降低建設樂園的成本，小薑打算把樂園分為 $M$ 個園區。建設一個園區的成本是根據深度來計算的，假設園區內最深與最淺的深度分別為 $D_{\max}$ 和 $D_{\min}$，則該園區的建設成本為 $D_{\max} - D_{\min}$。小薑已經調查好了每個有點心的地方的深度，請你寫一個程式計算最小建設總成本是多少。

## ■ 輸入說明

輸入的第一行有一個正整數 $T(T \leq 40)$，代表測試資料的組數。

每一筆測試資料共有兩行。第一行有兩個整數 $N, M$ $(1 \leq M \leq N \leq 50000)$，中間以一個空白隔開，代表計畫中有 $N$ 個有點心的地方，並打算把樂園分為 $M$ 個園區。

第二行有 $N$ 個整數 $D_i$ $(1 \leq D_i \leq 10^9)$，各以一個空白隔開，代表第 $i$ 個有點心的地方的深度。

## ■ 輸出說明

對於每一筆測試資料請輸出一行，包含一個整數，表示建設樂園的最小總成本。

## ■ 註腳

1. 薑餅人不會吃壞肚子。
2. 有點心的地方，就是樂園。

## ■ 範例輸入

```
2
5 2
9 7 6 2 1
5 3
9 7 6 2 1
```

## ■ 範例輸出

```
4
2
```

# 題目 C
# 棒球練習場

**執行時間限制: 1 秒**

公元四八八八年，由於空間傳送裝置的普及，人類對於能夠將物品傳送至不同空間，早就習以為常，並且作了許多應用。

有一天，皮皮和球球決定到「眼明手快」棒球場上進行終極棒球的決鬥。這場決鬥的規則很簡單，打到最多球的人就贏了。不過有趣的事情是，當棒球出現在某個位置 $(x, y)$ 之後，下一秒，棒球就會被瞬間傳送到 $(x \oplus y, |x - y|)$ 的位置。其中 $\oplus$ 是把 $x$ 和 $y$ 寫成二進位以後進行 XOR 運算的意思。

舉個例子來說，一開始的球出現在 $(5, 3)$ 的位置，那麼過了一秒以後就會出現在 $(6, 2)$，再下一秒就會出現在 $(4, 4)$，第四秒就會在 $(0, 0)$ 了。

皮皮和球球經過了詳盡的觀察以後，發現了一件事：只要 $x = y$，下一秒球就會出現在 $(0, 0)$。於是，皮皮和球球打算在 $(0, 0)$ 這個位置守株待球，只要球一出現，就可以把它打出去！

請你幫忙算算，若現在球的初始位置 $(a, b)$ 滿足：$1 \le a \le N$ 且 $1 \le b \le M$，那麼有哪些位置的球經過一段時間之後就會出現在 $(0, 0)$ 呢？

## ■ 輸入說明

輸入的第一行有一個正整數 $T(T \le 50)$，代表測試資料的組數。

每一筆測試資料，包含兩個正整數 $N, M(1 \le N, M \le 10000)$，中間以一個空白隔開。

## ■ 輸出說明

對於每一筆測試資料請輸出一行，包含一個整數，表示有幾球會出現在 $(0, 0)$。

## ■ 範例輸入

```
3
3 4
2 5
514 217
```

## ■ 範例輸出

```
12
10
111538
```

# 題目 D
# 樣式圖像辨識

**執行時間限制: 5 秒**

就如同大部份的人所知道的，世界上存在著許多古文明，如兩河流域、埃及、中國等地區都有大型的古文明存在，而其中有些古文明是已經有文字概念的。

然而，大多的古代文字是難以被辨認的，更一般地而言，大部份的古代「文字」其實還是比較像「圖騰」，只是其中可能存在著一些有意義的規律，使得那些圖像可以被視為原始的文字。對此，有許多考古學家及文字學家想要借由大量採集的文本來分析出可能的規律，進而期望能夠找到每一個「文字」的意義。

在找尋規律的過程中，有些「基本樣式」已經被發現了。所謂的「基本樣式」，即為在許多文本中出現很多次的圖形樣式。這類基本樣式是學者們首先想要知道的，由於出現的頻率較高，基本樣式的意義往往也比較容易被解析，也時常會在往後成為分析整個語言的關鍵。

為了要分析各種基本樣式的意義，首先必須要找出這些基本樣式出現在哪些文本中。然而，一個文本可能包含許多圖騰樣式，而且書寫於其上的樣式也有可能經過旋轉，導致這件工作很難由人眼來完成。因此，我們希望能夠使用電腦程式來完成這件事。請你/妳寫一支程式，判斷一則文本中是否包含給定的基本樣式。

為了簡化題目，我們定義文本為一個 $R \times C$ 的矩形網格 (grid)，裡面的每一格皆為 '.' 或 '*' (皆不包含引號)。我們定義樣式為「由 '*' 所組成的**連通塊**」。更精確地說，對於文本中的一個 '*'，其上、下、左、右四個方向相鄰的 '*' 屬於同一個樣式。

在要判斷的文本中，目標的基本樣式可能經過 90 度、180 度、270 度的旋轉，但不會經過縮放或翻轉。( 即便文本中存在一個經過縮放、翻轉後等於基本樣式的樣式，我們仍不認為該樣式為我們想要找的基本樣式)

給定的基本樣式則是一恰好包含一個樣式的文本。

## ■ 輸入說明

輸入的第一行有一個正整數 $T$ ($T \leq 50$)，代表測試資料的組數。

每一組測試資料分成兩個部分，第一個部分是目標的基本樣式 (以恰好包含一個樣式的文本呈現)，第二個部分則是給定的文本。

每個部分的第一行包含兩個整數 $R$ ($1 \leq R \leq 1024$) 和 $C$ ($1 \leq C \leq 1024$)，代表文本的大小。之後有 $R$ 行，每行有 $C$ 個字元。每個字元不是 '.' 就是 '*' (皆不包含引號)。

## ■ 輸出說明

對於每一筆測試資料，請輸出一行。

如果給定的文本中存在目標的基本樣式，則請輸出 "Yes" (不包含雙引號)；否則，請輸出 "No" (不包含雙引號)。

## ■ 範例輸入

```
4
7 9
.........
..*****..
..*****..
..*****..
.........
.........
.........
5 8
*.......
*.****..
*.****..
*.****..
*.......
```

```
2 2
**
*.
4 4
.*..
*...
...*
..**
7 7
.......
..****.
..*..*.
..****.
..*....
..*....
.......
7 7
.......
..****.
..*..*.
..****.
.....*.
.....*.
.......
5 5
*****
*...*
*...*
*...*
*****
5 5
*****
*...*
*.*.*
*...*
*****
```

## ■ 範例輸出

```
Yes
Yes
No
Yes
```

# 題目 E
# 可魚果贈送問題

**執行時間限制: 5 秒**

身為一隻有愛心的可魚果商，有天你在路上販賣可魚果的時候遇到了一對可愛的可魚姐妹，忍不住想要送些可魚果給她們。

已知你有 $N$ 包可魚果，第 $i$ 包可魚果裡面有 $C_i$ 顆可魚果，為求公平起見，你希望可以挑若干包給姐姐，若干包給妹妹，使得姐妹倆得到的可魚果數量相同。請問有沒有辦法做到呢？

雖然你很大方，但是你不一定要把所有可魚果都送出去。

## ■ 輸入說明

輸入的第一行有一個正整數 $T(T \leq 1000)$，代表測試資料的組數。

每一組測試資料有兩行。第一行有一個正整數 $N$ $(N \leq 50000)$，第二行包含了 $N$ 個正整數 $C_i$ $(C_i \leq 50000)$。

保證至少 99% 的測試資料中，$N \leq 1000$。

## ■ 輸出說明

對於每一筆測試資料請輸出一行。

如果可行的話請輸出 “Yes”，否則請輸出 “No”。(皆不包含雙引號)

## ■ 範例輸入

```
3
3
1 2 3
3
1 2 4
3
1 1 1
```

## ■ 範例輸出

```
Yes
No
Yes
```

# 題目 F
# 可魚果滾動問題

**執行時間限制: 1 秒**

你是一隻住在可喵國的小可喵，身為一隻貓，你最喜歡的就是玩食物。最近你發現了一種很好玩的食物：可魚果，來自可魚國的一種平民美食。因為可魚國的製作技術高超，每顆可魚果都是個正圓球，且質量、體積也都一模一樣，放在盤子上滾來滾去異常平順而賞心悅目。

一天，當你又在操弄著盤中的可魚果時，你發現一件驚人的事實：可魚果相撞時會進行彈性碰撞！有話是獨樂樂不如眾樂樂，於是你邀請了你的可喵好朋友們來家中，好讓大家一起驚嘆可魚果的奧妙。你準備了一個 $N$ 邊形的大盤子，並讓你與你的朋友們貓手一顆可魚果，大家各在盤子上選定一個起始點後同時將自己的可魚果向某個方向推去，如此一來大家就可以一齊觀賞一堆可魚果撞來撞去的美妙風景。當然，為了要讓畫面更加的自然、和諧，所有貓推出去的可魚果滾動速率都是每秒 $e$ 喵分 (喵分為可喵國的官方**距離單位**)。

正當你們玩可魚果玩得不亦樂乎時，你注意到一件事情：當只有自己一隻貓在玩可魚果的時候，因為滾動的可魚果數量很少，自己總是能夠精準的判斷哪些可魚果將掉出盤子，並精準地將它接起來。而當貓多了起來，可魚果的碰撞狀況變得複雜，就難以再精準地掌握究竟自己推出去的可魚果何時會掉出盤子了！

為了避免大家因為漏接而疲於奔命地去撿可魚果，你決定開發一個程式，精準地算出每一顆可魚果掉出盤子的時間，好讓大家準備好接起自己所推出去的可魚果。雖然你不會寫程式，但你相信無限貓咪定理，只要持之以恆地在鍵盤上亂踩，你總有一天會寫出想要的程式。

然而，在認真地嘗試了七七四十九秒後，你發現這個方法有個巨大的問題：你不知道怎麼驗證程式的正確性！為此，你打算準備一些測試資料。你認為，只要程式能夠成功的算出最後一顆掉出盤子的可魚果是滾了多遠才掉下去，那麼程式十之八九就是對的。只是問題又來了，你該怎麼算出最後一顆掉出盤子的可魚果滾了**多遠**呢？

由於這個問題對可喵們來說實在太難了，你決定在計算這個問題的答案時**忽視所有可魚果之間的碰撞**。

## ■ 輸入檔說明

輸入的第一行有一個正整數 $T$ ($T \leq 30$)，代表測試資料的組數。

每一筆測試資料的第一行有兩個正整數 $N$ ($3 \leq N \leq 400$), $M$ ($M \leq 500$)，中間以一個空白隔開，代表盤子是一個 $N$ 邊的凸多邊形，你和你朋友總共有 $M$ 隻貓。

接下來的 $N$ 行每一行有兩個數字 $X_i, Y_i$ ($-1000 \leq X_i, Y_i \leq 1000$) ，中間以一個空白隔開，代表盤子的各個頂點的位置（依照順時針或逆時針順序）。

接下來 $M$ 行每行有四個數字 $PX_i, PY_i, DX_i, DY_i$ ($-1000 \leq DX_i, DY_i \leq 1000$, $DX_i^2 + DY_i^2 > 0$) 各以一個空白隔開，分別代表每一顆可魚果的起始位置與方向向量 (注意，這只用以表示方向，速率都是每秒 $e$ 喵分)。保證每顆可魚果剛開始都落在盤子上 (可能在邊上)。

測試資料中的所有座標都是以喵分為單位。因為盤子很大，可魚果可以被視作一個點，也可以沿著盤子的邊滾動。且盤子的摩擦力小到可被忽視。

## ■ 輸出檔說明

對於每一筆測試資料請輸出一行，包含一個數字代表在不考慮任何碰撞的情況下，最後一顆掉出盤子的可魚果滾了多少喵分。所有輸出需四捨五入至小數點下第四位，保證 $10^{-6}$ 以內的誤差不會造成輸出答案改變。

## ■ 註腳

但是這些測試資料並不是你用無限貓咪定理踩出來的。

## ■ 範例輸入

```
3
3 1
0.0 0.0
3.0 0.0
0.0 2.0
1.0 1.0 0.0 1.0
4 2
0.0 0.0
0.0 2.0
2.0 2.0
2.0 0.0
1.0 1.0 1.0 0.0
2.0 0.0 0.0 1.0
4 2
0.0 0.0
2.0 0.0
2.0 2.0
0.0 2.0
0.0 2.0 0.0 1.0
2.0 0.0 1.0 0.0
```

## ■ 範例輸出

```
0.3333
2.0000
0.0000
```

*本頁留白。*

# 題目 G
# 胖胖天大大薯 II

**執行時間限制: 5 秒**

胖胖天為了達成他天天胖的野望，每天都以吃垮胖胖天國的麥當當為目標生活著。吃著吃著竟發現驚人的事實——胖胖天國內的麥當當同一天裡大薯的量是固定的！也就是說，同一天裡無論早晚拿到的一包大薯裡面的薯條根數都一樣，真是辛苦店員了。

為免釀成一天吃一百包大薯的悲劇，胖胖天決定一天最多只買一次大薯，但是他可以選擇在正常時段升級套餐吃**一包大薯**，或者利用晚上十點後買大送大的優惠一天吃**兩包大薯**，又或者他可以選擇那天就颯爽**不吃大薯**，就算點套餐也要把薯條換成玉米濃湯（他才不吃中薯小薯什麼的，一口就沒了太小家子氣了）。

隨著時間過去，胖胖天覺得大薯的量變少實在太不開心了！因此，如果胖胖天某天吃了 $X$ 根薯條，那從此以後如果有吃薯條的話，該天至少也要吃 $X$ 根薯條。

好心的正妹店員湯湯偷偷透露了接下來 $N$ 天麥當當的一包大薯有幾根薯條給胖胖天知道。已知聰明的胖胖天會吃儘量多天的大薯，且上次吃薯條時該天吃了 $M$ 根，請問他最多在接下來 $N$ 天中可以吃到幾天大薯呢？

## ■ 輸入說明

輸入的第一行有一個正整數 $T$ ($T \leq 1000$)，代表測試資料的組數。

每一組測試資料有兩行。第一行有兩個正整數 $N, M$ ($N \leq 100000, M \leq 10^9$)，中間以一個空白隔開。

第二行包含了 $N$ 個正整數 $C_i$ ($C_i \leq 10^9$) ，中間各以一個空白隔開，代表接下來第 $i$ 天時一包大薯會有 $C_i$ 根。

保證至少 99% 的測試資料中 $N \leq 1000$。

## ■ 輸出說明

對於每一筆測試資料請輸出一行，包含一個整數表示胖胖天在接下來 $N$ 天中最多可以吃幾天大薯。

## ■ 註腳

1. 胖胖天國的麥當當使用須彌芥子袋來裝大薯，不必擔心裝不下的問題。

2.「我可是靈活的胖子」胖胖天道。

## ■ 範例輸入

```
4
3 1
1 1 1
3 1
1 2 3
3 1
1 3 2
3 3
5 1 4
```

## ■ 範例輸出

```
3
3
3
2
```

# 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
```