實作題

第 1 題　三角形辨別

問題描述

三角形除了是最基本的多邊形外，亦可進一步細分為鈍角三角形、直角三角形及銳角三角形。若給定三個線段的長度，透過下列公式的運算，即可得知此三線段能否構成三角形，亦可判斷是直角、銳攬和鈍角三角形。

提示：若 a、b、c 為三個線段的邊長，且 c 為最大值，則

若 a + b ≦ c	，三線段無法構成三角形
若 a × a + b × b < c × c	，三線段構成鈍角三角形 (Obtuse triangle)
若 a × a + b × b = c × c	，三線段構成直角三角形 (Right triangle)
若 a × a + b × b > c × c	，三線段構成銳角三角形 (Acute triangle)

請設計程式以讀入三個線段的長度判斷並輸出此三線段可否構成三角形？若可，判斷並輸出其所屬三角形類型。

評分說明

輸入包含若干筆測試資料，每一筆測試資料的執行時間限制(time limit)均為 1 秒，依正確通過測資筆數給分。

輸入格式

輸入僅一行包含三正整數，三正整數皆小於 30,001，兩數之間有一空白。

輸出格式

輸出共有兩行，第一行由小而大印出此三正整數，兩數字之間以一個空白間格，最後一個數字後不應有空白；第二行輸出三角形的類型：

若無法構成三角形時輸出「No」；
若構成鈍角三角形時輸出「Obtuse」；
若直角三角形時輸出「Right」；
若銳角三角形時輸出「Acute」。

範例一：輸入

```
3 4 5
```

範例一：正確輸出

```
3 4 5
Right
```

（**說明**）a×a + b×b = c×c 成立時為直角三角形。

範例二：輸入

```
101 100 99
```

範例二：正確輸出

```
99 100 101
Acute
```

（**說明**）邊長排序由小到大輸出，a×a + b×b > c×c 成立時為銳角三角形。

範例三：輸入

```
10 100 10
```

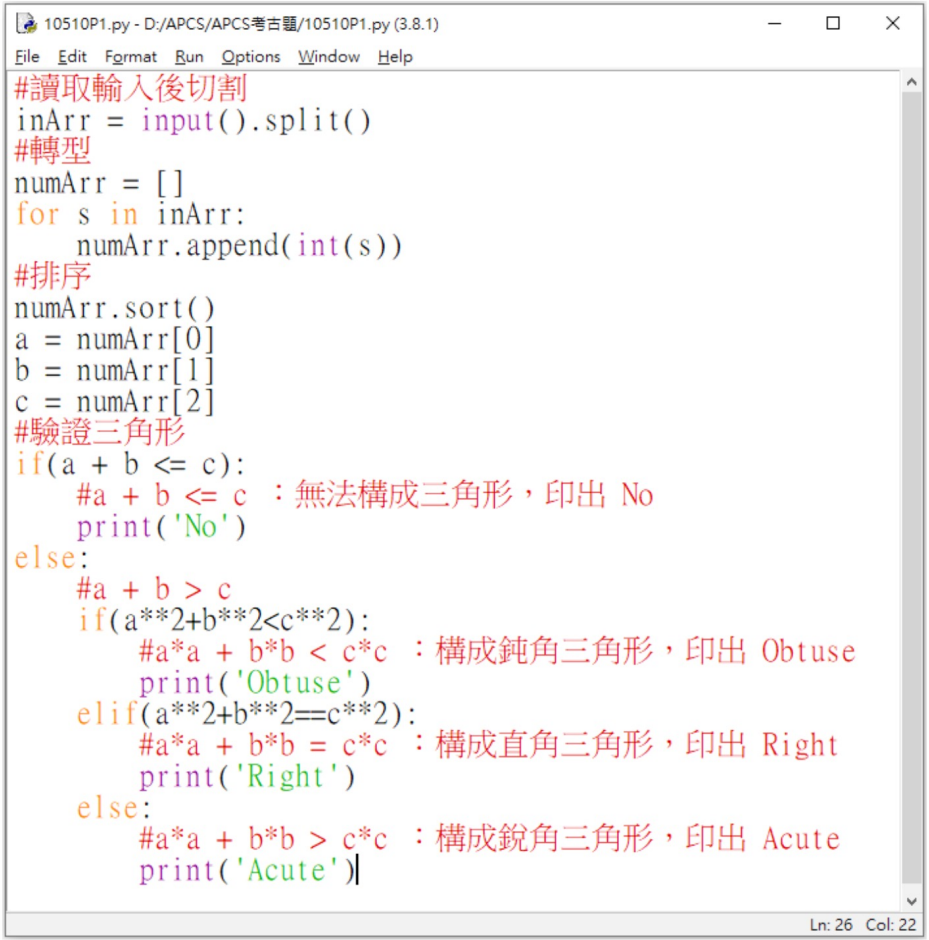
範例三：正確輸出

```
10 10 100
No
```

（**說明**）由於無法構成三角形，因此第二行須印出「No」。

解題思路

- 讀取輸入 a b c
  - 切割、轉型
  - 將 a b c 由小到大排序
- 驗證三角形
  - a + b <= c ：無法構成三角形，印出 No
  - a + b > c
    - a*a + b*b < c*c ：構成鈍角三角形，印出 Obtuse
    - a*a + b*b = c*c ：構成直角三角形，印出 Right
    - a*a + b*b > c*c ：構成銳角三角形，印出 Acute


10510P1.py - D:/APCS/APCS考古題/10510P1.py (3.8.1)
File Edit Format Run Options Window Help
#讀取輸入後切割
inArr = input().split()
#轉型
numArr = []
for s in inArr:
numArr.append(int(s))
#排序
numArr.sort()
a = numArr[0]
b = numArr[1]
c = numArr[2]
#驗證三角形
if(a + b <= c):
#a + b <= c ：無法構成三角形，印出 No
print('No')
else:
#a + b > c
if(a**2+b**2<c**2):
#a*a + b*b < c*c ：構成鈍角三角形，印出 Obtuse
print('Obtuse')
elif(a**2+b**2==c**2):
#a*a + b*b = c*c ：構成直角三角形，印出 Right
print('Right')
else:
#a*a + b*b > c*c ：構成銳角三角形，印出 Acute
print('Acute')
Ln: 26 Col: 22

實作題

第 2 題　最大和

問題描述

給定 N 群數字，每群都恰有 M 個正整數。若從每群數字中各選擇一個數字（假設第 i 群所選出數字為 t_i），將所選出的 N 個數字加總即可得總和 $S = t_1+t_2+\ldots+t_N$。請寫程式計算 S 的最大值（最大總和），並判斷各群所選出的數字是否可以整除 S。

評分說明

輸入包含若干筆測試資料，每一筆測試資料的執行時間限制(time limit)均為 1 秒，依正確通過測資筆數給分。其中：

第 1 子題組 20 分：$1 \leqq N \leqq 20$，$M = 1$。

第 2 子題組 30 分：$1 \leqq N \leqq 20$，$M = 2$。

第 3 子題組 50 分：$1 \leqq N \leqq 20$，$1 \leqq M \leqq 20$。

輸入格式

第一行有二個正整數 N 和 M， $1 \leqq N \leqq 20$，$1 \leqq M \leqq 20$。

接下來的 N 行，每一行各有 M 個正整數 x_i，代表一群整數，數字與數字間有一個空格，且 $1 \leqq i \leqq M$，以及 $1 \leqq x_i \leqq 256$。

輸出格式

第一行輸出最大總和 S。

第二行按照被選擇數字所屬群的順序，輸出可以整除 S 的被選擇數字，數字與數字間以一個空格隔開，最後一個數字後無空白；若 N 個被選擇數字都不能整除 S，就輸出-1。

範例一：輸入

```
3 2
1 5
6 4
1 1
```

範例一：正確輸出

```
12
6 1
```

（說明）挑選的數字依序是 5, 6, 1，總和 S=12。而此三數中可整除 S 的是 6 與 1，6 在第二群，1 在第 3 群所以先輸出 6 再輸出 1。注意，1 雖然也出現在第一群，但她不是第一群中挑出的數字，所以順序是先 6 後 1。

範例二：輸入

```
4 3
6 3 2
2 7 9
4 7 1
9 5 3
```

範例二：正確輸出

```
31
-1
```

（說明）挑選的數字依序是 6,9,7,9，總和 S=31。而此四數中沒有可整除 S 的，所以第二行輸出-1。

解題思路

- 讀取資料
  - N M：群 個
  - 讀取N次，切割為M筆資料，將每一群反向排序後，加入二維集合 Nums
- 處理資料
  - 取得每一群中的最大值：取索引0，加入LNums集合
  - 最大值(LNums集合)加總和為S
  - 最大值(LNums集合)中可將S整除的數字加入RNums 集合
- 輸出
  - S
  - 最大值中可將S整除的數字(RNums 集合)，無資料顯示-1

10510P2.py - D:\APCSClass3\Solutions\10510\10510P2.py (3.9.7)

File Edit Format Run Options Window Help

```
#讀取資料 N M ：群 個
ins = list(map(int, input().split()))
N = ins[0]
M = ins[1]

#讀取N次，切割為M筆資料
#將每一群反向排序後，加入二維集合 Nums
nums=[]
for i in range(N):
    row = list(map(int, input().split()))
    row.sort(reverse=True)
    nums.append(row)

#取得每一群中的最大值(索引0)，加入LNums集合
LNums = []
for row in nums:
    LNums.append(row[0])
#print(LNums)

#LNums集合加總和為S
S = sum(LNums)
```

Ln: 25 Col: 0

```
#LNums集合中可將S整除的數字加入RNums 集合
RNums = []
for l in LNums:
    if(S%l==0):
        RNums.append(l)

#輸出 S
print(S)

#最大值中可將S整除的數字(RNums 集合)
#無資料顯示-1
if(len(RNums)!=0):
    for r in RNums:
        print(r, end=' ')
    print()
else:
    print(-1)
```

Ln: 20 Col: 13

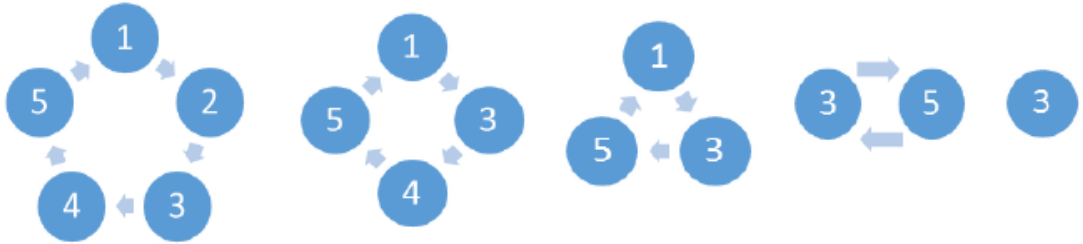
實作題

第 3 題　定時 K 彈

問題描述

「定時 K 彈」是一個團康遊戲，N 個人圍成一個圈，由 1 號依序到 N 號，從 1 號開始依序傳遞一枚玩具炸彈，炸彈每次到第 M 個人就會爆炸，此人即淘汰，被淘汰的人要離開圓圈，然後炸彈再從該淘汰者的下一個開始傳遞。遊戲之所以稱 K 彈是因為這枚炸彈只會爆炸 K 次，在第 K 次爆炸後，遊戲即停止，而此時在第 K 個淘汰者的下一位遊戲者被稱為幸運者，通常就會被要求表演節目。例如 N=5，M=2，如果 K=2，炸彈會爆炸兩次，被爆炸淘汰的順序依序是 2 與 4（參見下圖），這時 5 號就是幸運者。如果 K=3，剛才的遊戲會繼續，第三個淘汰的是 1 號，所以幸運者是 3 號。如果 K=4，下一輪淘汰 5 號，所以 3 號是幸運者。

給定 N、M 與 K，請寫程式計算出誰是幸運者。



輸入格式

輸入只有一行包含三個正整數，依序為 N、M 與 K，兩數中間有一個空格分開。其中 $1 \le K < N$。

輸出格式

請輸出幸運者的號碼，結尾有換行符號。

範例一：輸入

```
5 2 4
```

範例一：正確輸出

```
3
```

（說明）

被淘汰的順序是 2、4、1、5，此時 5 的下一位是 3，也是最後剩下的，所以幸運者是 3。

範例二：輸入

```
8 3 6
```

範例二：正確輸出

```
4
```

（說明）

被淘汰的順序是 3、6、1、5、2、8，此時 8 的下一位是 4，所以幸運者是 4。

評分說明

輸入包含若干筆測試資料，每一筆測試資料的執行時間限制(time limit)均為 1 秒，依正確通過測資筆數給分。其中：

第 1 子題組 20 分，$1 \le N \le 100$，且 $1 \le M \le 10$，$K = N\text{-}1$。

第 2 子題組 30 分，$1 \le N \le 10{,}000$，且 $1 \le M \le 1{,}000{,}000$，$K = N\text{-}1$。

第 3 子題組 20 分，$1 \le N \le 200{,}000$，且 $1 \le M \le 1{,}000{,}000$，$K = N\text{-}1$。

第 4 子題組 30 分，$1 \le N \le 200{,}000$，且 $1 \le M \le 1{,}000{,}000$，$1 \le K < N$。

- 讀取輸入 N M K
  - 切割、轉型
  - 建立pList，索引值為1-N
  - 索引idx由0開始
  - 執行遊戲K次
    - 每次遊戲傳遞炸彈M次爆炸，索引為(idx+M-1)%len(pList)
    - 彈出爆炸的玩家， pList.pop(idx)
  - 取出目前索引指向的物件
    - pList[idx%len(pList)]

10510P3-2.py - D:\APCSClass3\Solutions\10510\10510P3-2.py (3.9.7)

File Edit Format Run Options Window Help

```
def josephusLuckyPerson(ls, m, k):
    idx = 0
    for i in range(k):
        idx = (idx+m-1)%len(ls)
        ls.pop(idx)
    return ls[idx%len(ls)]


#讀取輸入 N M K, 切割、轉型
inStrs = input().split(' ')
N = int(inStrs[0])
M = int(inStrs[1])
K = int(inStrs[2])

#建立pList，索引值為1-N
pList=[i for i in range(1, N+1)]
#print(pList)

luckyPerson = josephusLuckyPerson(pList, M, K)
print(luckyPerson)
```

Ln: 21 Col: 0

實作題

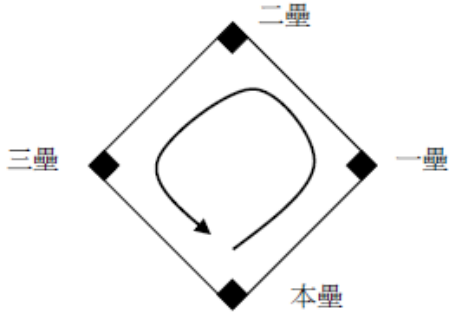
第 4 題　棒球遊戲

問題描述

謙謙最近迷上棒球，他想自己寫一個簡化的棒球遊戲計分程式。這個程式會讀入球隊中每位球員的打擊結果，然後計算出球隊的得分。

這是個簡化版的模擬，假設擊球員的打擊結果只有以下情況：

(1) 安打：以 1B, 2B, 3B 和 HR 分別代表一壘打、二壘打、三壘打和全（四）壘打。
(2) 出局：以 FO, GO, 和 SO 表示。



請寫出具備這樣功能的程式，計算球隊的總得分。

這個簡化版的規則如下：

(1) 球場上有四個壘包，稱為本壘、一壘、二壘和三壘。
(2) 站在本壘握著球棒打球的稱為「擊球員」，站在另外三個壘包的稱為「跑壘員」。
(3) 當擊球員的打擊結果為「安打」時，場上球員（擊球員與跑壘員）可以移動；結果為「出局」時，跑壘員不動，擊球員離場，換下一位擊球員。
(4) 球隊總共有九位球員，依序排列。比賽開始由第 1 位開始打擊，當第 i 位球員打擊完畢後，由第 (i+1) 位球員擔任擊球員。當第九位球員完畢後，則輪回第一位球員。
(5) 當打出 K 壘打時，場上球員（擊球員和跑壘員）會前進 K 個壘包。從本壘前進一個壘包會移動到一壘，接著是二壘、三壘，最後回到本壘。
(6) 每位球員回到本壘時可得 1 分。
(7) 每達到三個出局數時，一、二和三壘就會清空（跑壘員都得離開），重新開始。

輸入格式

1. 每組測試資料固定有十行。
2. 第一到九行，依照球員順序，每一行代表一位球員的打擊資訊。每一行開始有一個正整數 a $(1 \le a \le 5)$，代表球員總共打了 a 次。接下來有 a 個字串（均為兩個字元），依序代表每次打擊的結果。資料之間均以一個空白字元隔開。球員的打擊資訊不會有錯誤也不會缺漏。
3. 第十行有一個正整數 b $(1 \le b \le 27)$，表示我們想要計算當總出局數累計到 b 時，該球隊的得分。輸入的打擊資訊中至少包含 b 個出局。

輸出格式

計算在總計第 b 個出局數發生時的總得分，並將此得分輸出於一行。

評分說明

輸入包含若干筆測試資料，每一筆測試資料的執行時間限制(time limit)均為 1 秒，依正確通過測資筆數給分。其中：

第 1 子題組 20 分，打擊表現只有 HR 和 SO 兩種。

第 2 子題組 20 分，安打表現只有 1B，而且 b 固定為 3。

第 3 子題組 20 分， b 固定為 3。

第 4 子題組 40 分，無特別限制。

範例一：輸入

```
5 1B 1B FO GO 1B
5 1B 2B FO FO SO
4 SO HR SO 1B
4 FO FO FO HR
4 1B 1B 1B 1B
4 GO GO 3B GO
4 1B GO GO SO
4 SO GO 2B 2B
4 3B GO GO FO
3
```

範例一：正確輸出

```
0
```

（說明）
1B：一壘有跑壘員。
1B：一、二壘有跑壘員。
SO：一、二壘有跑壘員，一出局。
FO：一、二壘有跑壘員，兩出局。
1B：一、二、三壘有跑壘員，兩出局。
GO：一、二、三壘有跑壘員，三出局。

達到第三個出局數時，一、二、三壘均有跑壘員，但無法得分。因為 $b=3$，代表三個出局就結束比賽，因此得到 0 分。

範例二：輸入

```
5 1B 1B FO GO 1B
5 1B 2B FO FO SO
4 SO HR SO 1B
4 FO FO FO HR
4 1B 1B 1B 1B
4 GO GO 3B GO
4 1B GO GO SO
4 SO GO 2B 2B
4 3B GO GO FO
6
```

範例二：正確輸出

```
5
```

（說明）接續範例一，達到第三個出局數時未得分，壘上清空。
1B：一壘有跑壘員。
SO：一壘有跑壘員，一出局。
3B：三壘有跑壘員，一出局，得一分。
1B：一壘有跑壘員，一出局，得兩分。
2B：二、三壘有跑壘員，一出局，得兩分。
HR：一出局，得五分。
FO：兩出局，得五分。
1B：一壘有跑壘員，兩出局，得五分。
GO：一壘有跑壘員，三出局，得五分。

因為 $b=6$，代表要計算的是累積六個出局時的得分，因此在前 3 個出局數時得 0 分，第 4~6 個出局數得到 5 分，因此總得分是 0+5=5 分。

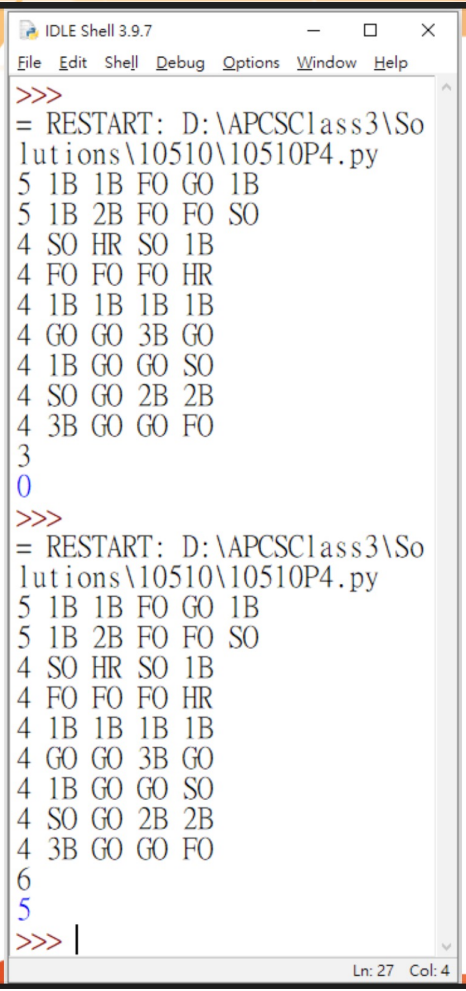
```python
class GameOver(Exception):
    pass

records = [[] for _ in range(9)]
for i in range(9):
    record = input().split()
    for j in range(1, int(record[0])+1):
        records[i].append(record[j])
out = int(input())
outCount = 0
bases = [0,0,0]
outs = ['FO','GO','SO']
scores, now = 0, 0
try :
    for i in range(len(records[0])):
        for j in range(9):
            if records[j][i] in outs:
                outCount+=1
                now+=1
            elif records[j][i]=='1B':
                scores += bases.pop(0)
                bases.append(1)
            elif records[j][i]=='2B':
                scores += bases.pop(0)
                scores += bases.pop(0)
                bases.append(1)
                bases.append(0)
            elif records[j][i]=='3B':
                scores += bases.pop(0)
                scores += bases.pop(0)
                scores += bases.pop(0)
                bases.append(1)
                bases.append(0)
            elif records[j][i]=='HR':
                scores += bases.pop(0)
                bases.append(1)
                scores += bases.pop(0)
                scores += bases.pop(0)
                scores += bases.pop(0)
                bases.append(0)
                bases.append(0)
                bases.append(0)
            if now==3:
                bases = [0,0,0]
                #out-=3
                now=0
            #print(records[j][i], bases, scores,
            if outCount == out:
                raise GameOver()
except:
    print(scores)
```

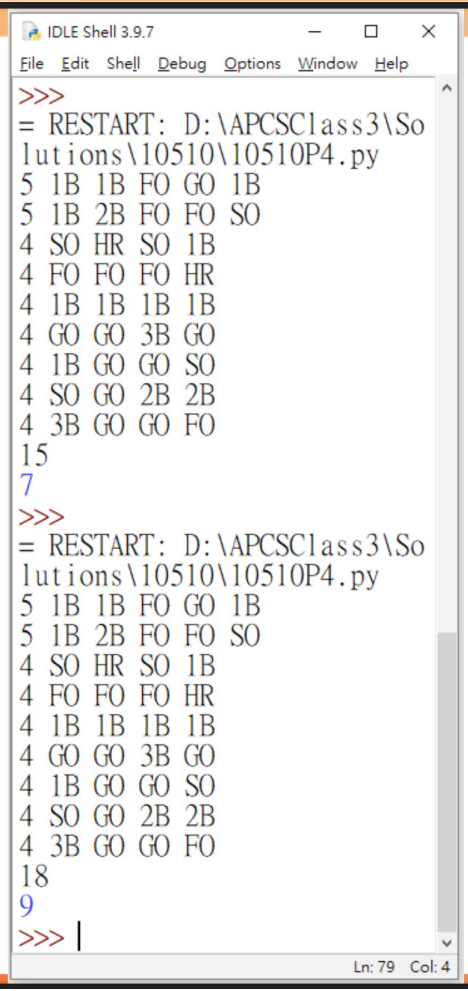
範例二：輸入

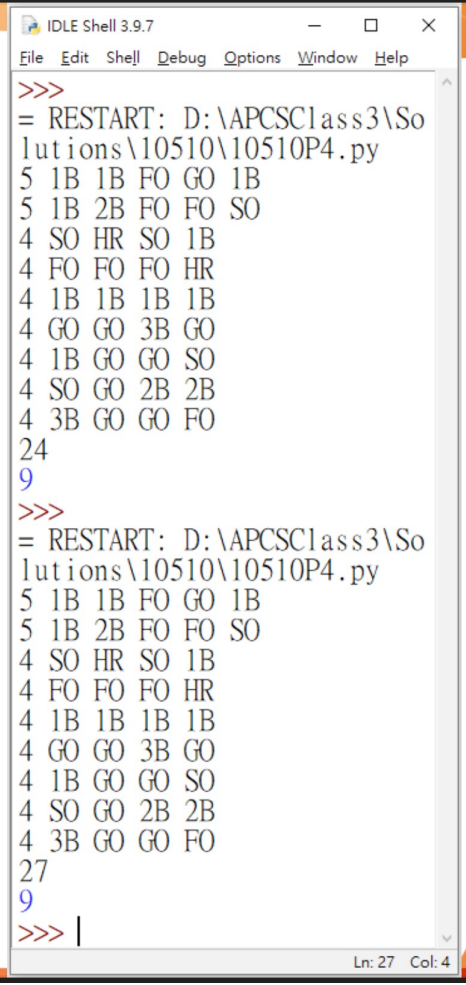
```
5 1B 1B FO GO 1B
5 1B 2B FO FO SO
4 SO HR SO 1B
4 FO FO FO HR
4 1B 1B 1B 1B
4 GO GO 3B GO
4 1B GO GO SO
4 SO GO 2B 2B
4 3B GO GO FO
6
```

範例二：正確輸出

```
5
```


IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
>>>
= RESTART: D:\APCSClass3\Solutions\10510\10510P4.py
5 1B 1B FO GO 1B
5 1B 2B FO FO SO
4 SO HR SO 1B
4 FO FO FO HR
4 1B 1B 1B 1B
4 GO GO 3B GO
4 1B GO GO SO
4 SO GO 2B 2B
4 3B GO GO FO
3
0
>>>
= RESTART: D:\APCSClass3\Solutions\10510\10510P4.py
5 1B 1B FO GO 1B
5 1B 2B FO FO SO
4 SO HR SO 1B
4 FO FO FO HR
4 1B 1B 1B 1B
4 GO GO 3B GO
4 1B GO GO SO
4 SO GO 2B 2B
4 3B GO GO FO
6
5
>>>
Ln: 27 Col: 4


IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
>>>
= RESTART: D:\APCSClass3\Solutions\10510\10510P4.py
5 1B 1B FO GO 1B
5 1B 2B FO FO SO
4 SO HR SO 1B
4 FO FO FO HR
4 1B 1B 1B 1B
4 GO GO 3B GO
4 1B GO GO SO
4 SO GO 2B 2B
4 3B GO GO FO
15
7
>>>
= RESTART: D:\APCSClass3\Solutions\10510\10510P4.py
5 1B 1B FO GO 1B
5 1B 2B FO FO SO
4 SO HR SO 1B
4 FO FO FO HR
4 1B 1B 1B 1B
4 GO GO 3B GO
4 1B GO GO SO
4 SO GO 2B 2B
4 3B GO GO FO
18
9
>>>
Ln: 79 Col: 4


IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
>>>
= RESTART: D:\APCSClass3\Solutions\10510\10510P4.py
5 1B 1B FO GO 1B
5 1B 2B FO FO SO
4 SO HR SO 1B
4 FO FO FO HR
4 1B 1B 1B 1B
4 GO GO 3B GO
4 1B GO GO SO
4 SO GO 2B 2B
4 3B GO GO FO
24
9
>>>
= RESTART: D:\APCSClass3\Solutions\10510\10510P4.py
5 1B 1B FO GO 1B
5 1B 2B FO FO SO
4 SO HR SO 1B
4 FO FO FO HR
4 1B 1B 1B 1B
4 GO GO 3B GO
4 1B GO GO SO
4 SO GO 2B 2B
4 3B GO GO FO
27
9
>>>
Ln: 27 Col: 4