

## C++ Test

2022 機器人實驗室 C++測驗 出題者：吳政彥

### Outline

#### Simple

[Q1 – GPA Calculator](#)

[Q2 – Vocabulary 7000](#)

#### Medium

[Q3 – Pointer](#)

[Q4 – Bowling](#)

#### Hard (Optional)

[Q5 – Simple Calculator \(Stack\)](#)

[Q6 – McDonald's \(Queue\)](#)

[Usage of testcase](#)

[Submission Format](#)

# Q1 – GPA Calculator

本學期快結束，Offliner 已經知道自己各科的學期總成績，想知道自己這學期的 GPA 是多少，請各位幫 Offliner 寫一個 C++ 的程式來計算他的 GPA 是多少。

下表為百分制與等第制對照表

百分制	等第制
90 - 100	A+ (4.3)
85 - 89	A (4)
80 - 84	A- (3.7)
77 - 79	B+ (3.3)
73 - 76	B (3)
70 - 72	B- (2.7)
67 - 69	C+ (2.3)
63 - 66	C (2)
60 - 62	C- (1.7)
0 - 59	F (0)

## Input

第一個整數 M 為 Offliner 這學期修課數量(M 不會超過 int 上界)，接著會接續 M 行資料，每行資料會有兩個數字，第一個為該科目的百分制分數(浮點數)，第二個為該科目的學分數(整數)，這兩個數字之間會用空白隔開，其中該科目若不及格則會顯示 F，若該科目停修則會顯示 W，且停修科目不列入 GPA 計算。科目的分數需先採四捨五入至整數再做 GPA 的權重計算。

## Output

輸出 Offliner 該學期的 GPA，該值須四捨五入至小數第二位，若不及格的學分數大於該學期總學分數(含停修學分數)的一半，則需換行印出 **flunk out**。

## Example 1

Input	Ouput
4 97 3 78 2 84 2 F 3	2.69

**Example 2**

Input	Ouput
7	2.35
91 2	
W 1	
81.5 2	
F 3	
78 2	
66.7 3	
59.5 2	

**Example 3**

Input	Ouput
5	1.77
84.6 2	
F 2	
81.5 2	
F 3	
69 1	

**Example 4**

Input	Ouput
3	1.6
F 3	flunk out
94.215 1	
83 1	

## Q2 – Vocabulary 7000

**Offliner** 在當高中生的英文家教時，遇到了一個的學生名叫小明，他非常不喜歡使用手機軟體來背 7000 單，因為軟體通常會內建許多廣告，還有需要付費內容，這容易使他十分不想使用，因此請你使用 **C++** 來撰寫能輸入英文單字後，並透過查詢 7000 單的 **csv** 資料表來找到其單字的詞性與中文意思。

7000 單的 **csv** 資料表有兩欄，第一欄為英文單字，第二欄為單字詞性與單字中文意思，單字詞性在前而單字中文意思在後。

### Input

首先會先輸入數字 **N**(**N** 不會超過 **int** 上界)，接下來會接續輸入 **N** 個英文單字，英文單字可能會摻雜大小寫，甚至是數字。

### Output

輸出 **N** 個單字的詞性與中文意思，兩者之間用一個空白隔開，若找不到該單字直接輸出 **Unknown**。

### Example 1

Input	Ouput
4	v. 吸收；理解
absorb	n. 反派角色
villain	n. 甲狀腺
thyroid	n./v. 汗水；出汗
sweat	

### Example 2

Input	Ouput
8	n./v. 爭論
Debate	n. 長袍
RoBE	adj. 對...不感興趣
indifFeReNt	Unknown
PnEuMonoultramicRoscopiCsilicOVOLcanoconiosis	adj. 理論(上)的
THEORETICAL	n. 愛國者
patRioT	n. 護照
paSSpoRt	Unknown
b00k	

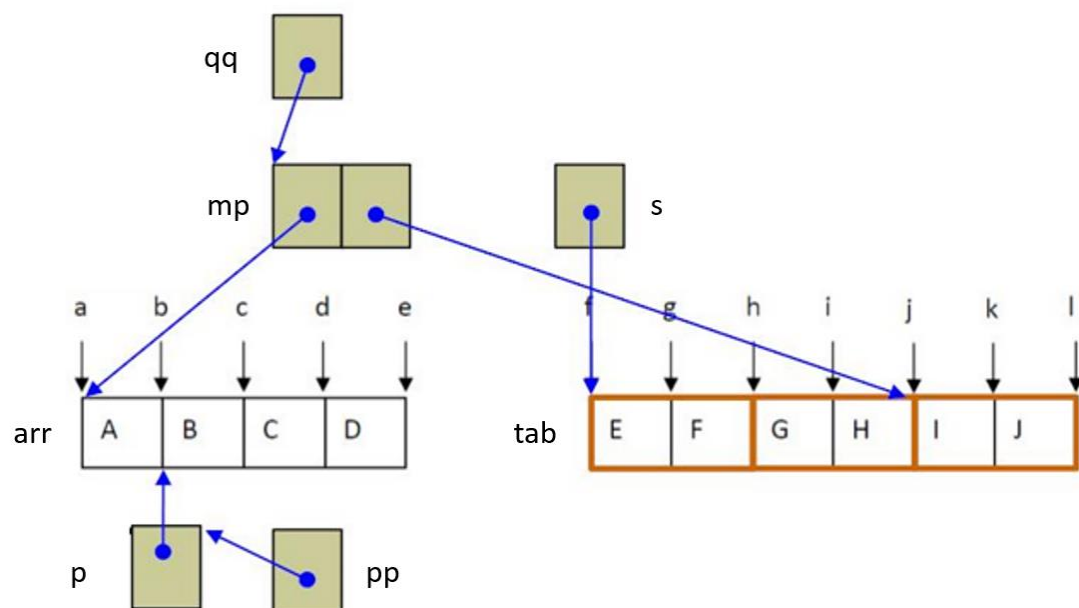
# Q3 – Pointer

由於 Offliner 苦手於 C/C++ 的指標，不過他已經畫好該程式的指標圖，也完成表格的一部分，所以請你幫幫他完成剩下的部分吧(跪

## Code

```
1. int main()
2. {
3.     int arr[4];
4.     int tab[3][2];
5.     int *p = arr + 1;
6.     int *mp[2] = {arr, tab[2]};
7.     int **pp = &p;
8.     int **qq = mp;
9.     int (*s)[2] = tab;
10. }
```

## Pointer Graph



### Point table

請完成表格剩下的部分

Expression	Address or variable	Data type
arr	{A, B, C, D}	int [4]
arr[0]	A	int
arr+1	b	int *
tab+1	h	int (*)[2]
tab[1]	{G, H}	int [2]
tab[1]+1		
tab[1][1]		
&tab[1][1]		
&tab[1][1]+1		
p+0		
p+1		
*p		
p[2]		
mp[0][1]		
mp[1][1]		
*pp+2		
qq[1]		
qq[1]+1		
qq[0][0]		
qq[1][0]		
s[0][1]		
s[1][0]		

※Hint：其實也可以把陣列填滿不同值，再把每個位置的記憶體值記下來，最後直接把未完成的 **Expression** 給印出來看看

## Q4 – Bowling

最近因為實驗室有人確診，不幸受到波及的 Offliner 只能乖乖進行 3+4 隔離，不幸中的大幸是 Nintendo Switch 推出了 Sports 這款遊戲，而 Offliner 深陷在保齡球這運動中



但保齡球的計分方式有些複雜，就由 Offliner 娓娓道來。首先可以看到下圖為保齡球 10 局的計分表

1	2	3	4	5	6	7	8	9	10

每局有兩次的出手機會(第 10 局例外)，假如第一局的第二次出手擊倒 8 瓶則會記為

1	2	3	4	5	6	7	8	9	10
8									

若第一局的第二次出手只擊倒 1 瓶，則會記為

1	2	3	4	5	6	7	8	9	10
8	1								

接著就可以計算第 1 局的得分為 9，會記在第 1 局的下方

1	2	3	4	5	6	7	8	9	10
8	1								
9									

接著，如果第 2 局的第一次出手擊倒 7 瓶，而第二次出手把剩下的 3 瓶都擊倒，則稱為 Spare(Spare 會註記/)。發生 Spare 時會將上局的得分加上本局得滿分(10 分)，再加上下一次出手的分數當成獎勵分，所以第 2 局發生 Spare 時會將上一局的 9 分加上第 2 局的滿分 10 分，再加上第 3 局的第一次出手 9 分，最後第 2 局的計分為 28

1	2	3	4	5	6	7	8	9	10
8	1	7	/						
9	28								

若第 3 局又發生 Spare，會將第二局的 28 分加上本局滿分 10，再加上下一局的第一次出手 7 分，所以第 3 局會註記 45

1	2	3	4	5	6	7	8	9	10
8	1	7	/	9	/	7			
9	28	45							

第 4 局因為沒有把剩下的球瓶都擊倒，所以第 4 局只會由第 3 局的分數加上本局得分 9 分，因此第 4 局為 54 分

1	2	3	4	5	6	7	8	9	10
8	1	7	/	9	/	7	2		
9	28	45	54						

而當第 5 局一次出手就全倒，稱為 Strike(Strike 會標記 x)，Strike 可以獲得之後兩次出手的分數，所以第 4 局的 54 分，加上本局滿分 10 分，再加上之後一次出手的 10 分，還有之後第 2 次出手的 7 分，最後第五局記分為 81 分

1	2	3	4	5	6	7	8	9	10
8	1	7	/	9	/	7	2	x	
9	28	45	54	81					



第 6 局又發生 **Strike**，分數會由第 5 局的 81 加上第 6 局得滿分 10 分，在加上後兩次出手的得分，分別為 7 分與 2 分，所以第 6 局會註記為 100 分

1	2	3	4	5	6	7	8	9	10
8	1	7	/	9	/	7	2	x	
9	28	45	54	81	100				

依此類推，就可以算得遊玩圖片中的 164 分。而第 10 局比較特別，最多有 3 次的出手機會，但如果在第 10 局的前兩次出手沒能擊倒所有球瓶則會喪失第三次的出手機會，如

10
7
2

而第 10 局的第二次出手即使發生了 **Strike** 或 **Spare** 也不會有獎勵分數。為了程式方便，只要是該局 **Strike** 不用第 2 次出手，或者第 10 局喪失出手機會都會註記-，如下圖兩種情況

10	6
7	x
2	-
-	100

既然已經知道保齡球的計分規則，請幫幫 **Offliner** 來計算他的總分與他的積分，**Nintendo Switch Sports** 只要有參加就有 40 積分，當打出一次 **Spare** 會乘上 1.1，而打出 **Strike** 會乘上 1.2，如遊戲畫面中，**Offliner** 共擊出 3 次 **Strike** 與 4 次 **Spare**，所以能獲得  $40 \times 1.2 \times 1.2 \times 1.2 \times 1.1 \times 1.1 \times 1.1 \times 1.1 \times 1.1 \times 1.1 \div 101$  積分

### Input

會輸入最多 21 個出手標記(第 1 到 9 局各 2 個，第 10 局 3 個，且分別用一個空白隔開)，由於 **Nintendo Switch Sports** 是採 16 人淘汰制，因此不一定會打滿 10 局，也可能輸入低於 21 個出手標記，例如 6 個，表示只打 3 局。

### Output

輸出最後總得分與積分，積分採四捨五入至整數

**Example 1**

Input	Output
$8\ 1\ 7\ /\ 9\ /\ 7\ 2\ x - x - 7\ 2\ x - 9\ /\ 7\ /\ 8$	164 101

**Example 2**

Input	Output
$x - x - x - x - x - x - x - x - x - x - x\ x\ x$	300 357

**Example 3**

Input	Output
$x - 9\ /\ x - 8\ /\ x - 7\ /\ x - 6\ /\ x - 1\ /\ x$	200 192

**Example 4**

Input	Output
$8\ /\ 7\ /\ 6\ /\$	43 53

## Q5 – Simple Calculator (Stack)

近期因實驗室助理的離職，因此 Offliner 要負責實驗室的報帳工作，但報帳的過程十分繁雜，一堆數字也讓 Offliner 看得眼花撩亂，當報上去的數字與發票上的不同又會被退件，因此 Offliner 決定撰寫一個簡單的計算機來計算。

首先介紹什麼是 Infix Expression，這種表示法適合人類觀看，如下式：

$$(1 + 2) * (5 - 3) * 6 / 5$$

但這種表示法對電腦卻十分不友善，電腦適合的是 Postfix Expression，也是所謂的后序表示法，這種表示法會將運算子(加減乘除取餘等)放置運算符(1~9 等)的后方，上述的式子使用 Postfix Expression 就會變成

$$12 + 53 - * 6 * 5 /$$

當有了 Postfix Expression 後，就可以使用資料結構中的 Stack 來完成計算

Postfix	Stack (Right is top)	Output
12+53-*6*5/		
2+53-*6*5/	1	
+53-*6*5/	1,2	
53-*6*5/	3	
3-*6*5/	3,5	
-*6*5/	3,5,3	
*6*5/	3,2	
6*5/	6	
*5/	6,6	
5/	36	
/	36,5	
	7.2	
		7.2

透過上表可以看到，將 Postfix Expression 一個一個 push 輸入到 Stack 中，當遇到運算子時，會將 Stack 中最上層 top 的兩個 pop 出來做運算後再 push 加入 Stack，最後 Postfix Expression 都處理好後就將 Stack 的數字 pop 出來就是答案！

但問題是，Offliner 在使用計算機不是使用 Postfix Expression，而是 Infix Expression，因此需要再另一個 Stack 來做 Infix-Postfix 的轉換，假設有個 Infix Expression 是

$$(1 + 2) * (5 - 3) * 6 / 5$$

Infix	Stack (Right is top)	Postfix
(1+2)*(5-3)*6/5		
1+2)*(5-3)*6/5	(	
+2)*(5-3)*6/5	(	1
2)*(5-3)*6/5	(,+	1
)*(5-3)*6/5	(,+	12
*(5-3)*6/5		12+
(5-3)*6/5	*	12+
5-3)*6/5	*,(	12+
-3)*6/5	*,(	12+5
3)*6/5	*,(-	12+5
)*6/5	*,(-	12+53
*6/5	*	12+53-
6/5	*	12+53-*
/5	*	12+53-*6
5	/	12+53-*6*
	/	12+53-*6*5
		12+53-*6*5/

透過上表可以看到，將 Infix Expression 一個一個 push 輸入到 Stack 中，如果遇到數字直接加入 Postfix Expression，遇到運算符(除了右括號)則會先判斷 Stack 的 top 是不是\*，/或者%這種高優先執行的，如果是的話就將這些 pop 到 Postfix，再將運算符 push 輸入到 Stack 中，如果不是的話就直接 push 輸入到 Stack。

若遇到右括號，直接將 Stack 一直 pop 出來到 Postfix 中直到 Stack 的 top 是左括號為止。

現在已經知道如何從 Infix 轉換到 Postfix，且也知道怎麼用 Postfix Expression 作運算，請你幫幫 Offliner 使用 C++來撰寫一個簡單的計算機，讓 Offliner 的報帳效率變更高吧!

### Input

輸入一串 Infix Expression，並要能處理+,-,\*,/與%這五種運算符，要記得有括號會影響計算的優先順序，為了使答案都是整數，所以這裡/的計算結果要記得下高斯符號:

$$\frac{a}{b} = \left\lfloor \frac{a}{b} \right\rfloor$$

計算過程中可能會產生負數，但測資中對於所有的%運算都一定會保持兩正整數

### **Output**

當 Infix Expression 遇到=時，就要輸出=之前的計算結果，也就是說幾個等於就要輸出幾個計算結果

#### **Example 1**

Input	Output
1+3-2*4=	-4

#### **Example 2**

Input	Output
1+2*(3+(8-6/3)*5)=	67

#### **Example 3**

Input	Output
1+(2*3*(1+2+3*4/2*7+(1+8))+4)=/(11/5)+7- 19*2=	329 133

#### **Example 4**

Input	Output
1+2=*5+(1+7+11)=%5-11=	3 34 -7

## Q6 – McDonald's (Queue)

前陣子麥當勞推出了四喜餐，五福餐等優惠，一個原價 226 的套餐居然只需要 150 元就可以買到，連冰炫風只要加 1 元就能多一件



因此台大的活大麥當勞在用餐時段總是能看到排到外面的隊伍等著訂餐，而喜歡吃雙層牛肉吉士堡的 Offliner 當然不會放過這次機會。

然而有些不遵守排隊規則的同學常會自動「解壓縮」，也就是幫朋友排隊，等朋友來時自動插隊。

有些同學也會衡量排隊的時間成本，排一下就自動離隊去買一旁的素食。

而有時櫃檯的服務人員會因為需要內場需要炸薯條等而進廚房。

因此請你 C++ 來撰寫程式來顯示最後的排隊情況！

### Input

首先會輸入 M、N 與 K 三個整數(皆不會超過 int 的上界)，M 表示有幾個櫃台的服務人員，N 表示接下來會輸入 N 種情況，K 表示排隊的人有 K 個團體(即使是單人排隊，如 Offliner 這種沒朋友的，也會算一個團體)。

輸入的情況包含以下 4 種:

**go m** : 表示 m 號櫃台服務人員幫客人點好餐，所以排隊的隊伍所有人會往前一個

**enter i j m** : 表示 m 號櫃台的隊伍又多加一人排隊(加在隊伍末端), i 表示這個客人屬於的團體, j 表示這位客人的編號，當排隊隊伍前面也有團體 i 的人，則這位客人就不會加入在隊伍末端，而是該團體的末端

**leave m** : 表示 m 號櫃台的隊伍最後一人因為不想等太久，所以自動離隊

**close m** : 表示 m 號櫃台的服務人員因為要幫忙內場，所以該櫃臺不再服務，而排該櫃臺的隊伍會自動找附近還有服務的櫃台(往 m 編號小的開始找)並加入在該隊伍末端，要記得加入到另一個隊伍時是原本隊伍的最後一人會先加入到另一個隊伍的末端，而如果另一個隊伍又有屬於相同團體的同學時要記得插隊的情況

### **Output**

輸出每個櫃台最後的排隊情況，如果櫃台人員進內場 **close** 時，雖然該櫃臺沒有排隊的人，但也要記得印出來，所以一開始有 **M** 個櫃台服務人員，最後輸出也要是 **M** 行，輸出只要輸出客人編號就好

### **Example 1**

Input	Output
1 11 4 enter 0 1 0 enter 1 2 0 enter 0 3 0 enter 1 4 0 enter 2 5 0 enter 1 6 0 leave 0 leave 0 go 0 go 0 enter 0 7 0	2 4 7

有 1 個服務櫃台，11 個情況，與最多 4 個團體，客人會用下標來做標記方便解釋，數字為客人的編號，下標為該客人屬於的團體，而紅色表示插隊發生

情況	排隊情況
enter 0 1 0	0 號櫃台 : $1_0$
enter 1 2 0	0 號櫃台 : $1_0$ 、 $2_1$
enter 0 3 0	0 號櫃台 : $1_0$ 、 $3_0$ 、 $2_1$
enter 1 4 0	0 號櫃台 : $1_0$ 、 $3_0$ 、 $2_1$ 、 $4_1$
enter 2 5 0	0 號櫃台 : $1_0$ 、 $3_0$ 、 $2_1$ 、 $4_1$ 、 $5_2$
enter 1 6 0	0 號櫃台 : $1_0$ 、 $3_0$ 、 $2_1$ 、 $4_1$ 、 $6_1$ 、 $5_2$
leave 0	0 號櫃台 : $1_0$ 、 $3_0$ 、 $2_1$ 、 $4_1$ 、 $6_1$
leave 0	0 號櫃台 : $1_0$ 、 $3_0$ 、 $2_1$ 、 $4_1$
go 0	0 號櫃台 : $3_0$ 、 $2_1$ 、 $4_1$
go 0	0 號櫃台 : $2_1$ 、 $4_1$
enter 0 7 0	0 號櫃台 : $2_1$ 、 $4_1$ 、 $7_0$

### Example 2

Input	Output
3 14 4 enter 0 1 0 enter 1 2 0 enter 0 3 0 enter 2 4 1 enter 1 5 1 enter 0 6 1 enter 1 7 1 enter 3 8 1 enter 1 9 2 enter 0 10 2 close 1 go 0 close 0 go 2	5 7 2 10 6 3 4 8

有 3 個服務櫃台，14 個情況，與最多 4 個團體

情況	排隊情況
enter 0 1 0	0 號櫃台 : $1_0$ 1 號櫃台 : 2 號櫃台 :
enter 1 2 0	0 號櫃台 : $1_0$ 、 $2_1$ 1 號櫃台 :



	2 號櫃台：
enter 0 3 0	0 號櫃台： $1_0$ 、 $3_0$ 、 $2_1$ 1 號櫃台： 2 號櫃台：
enter 2 4 1	0 號櫃台： $1_0$ 、 $3_0$ 、 $2_1$ 1 號櫃台： $4_2$ 2 號櫃台：
enter 1 5 1	0 號櫃台： $1_0$ 、 $3_0$ 、 $2_1$ 1 號櫃台： $4_2$ 、 $5_1$ 2 號櫃台：
enter 0 6 1	0 號櫃台： $1_0$ 、 $3_0$ 、 $6_0$ 、 $2_1$ 1 號櫃台： $4_2$ 、 $5_1$ 2 號櫃台：
enter 1 7 1	0 號櫃台： $1_0$ 、 $3_0$ 、 $6_0$ 、 $2_1$ 1 號櫃台： $4_2$ 、 $5_1$ 、 $7_1$ 2 號櫃台：
enter 3 8 1	0 號櫃台： $1_0$ 、 $3_0$ 、 $6_0$ 、 $2_1$ 1 號櫃台： $4_2$ 、 $5_1$ 、 $7_1$ 、 $8_3$ 2 號櫃台：
enter 1 9 2	0 號櫃台： $1_0$ 、 $3_0$ 、 $6_0$ 、 $2_1$ 1 號櫃台： $4_2$ 、 $5_1$ 、 $7_1$ 、 $8_3$ 2 號櫃台： $9_1$
enter 0 10 2	0 號櫃台： $1_0$ 、 $3_0$ 、 $6_0$ 、 $2_1$ 1 號櫃台： $4_2$ 、 $5_1$ 、 $7_1$ 、 $8_3$ 2 號櫃台： $9_1$ 、 $10_0$
close 1	0 號櫃台： $1_0$ 、 $3_0$ 、 $6_0$ 、 $2_1$ 、 $7_1$ 、 $5_1$ 、 $8_3$ 、 $4_2$ 1 號櫃台：(Closed) 2 號櫃台： $9_1$ 、 $10_0$
go 0	0 號櫃台： $3_0$ 、 $6_0$ 、 $2_1$ 、 $7_1$ 、 $5_1$ 、 $8_3$ 、 $4_2$ 1 號櫃台：(Closed) 2 號櫃台： $9_1$ 、 $10_0$
close 0	0 號櫃台：(Closed) 1 號櫃台：(Closed) 2 號櫃台： $9_1$ 、 $5_1$ 、 $7_1$ 、 $2_1$ 、 $10_0$ 、 $6_0$ 、 $3_0$ 、 $4_2$ 、 $8_3$
go 2	0 號櫃台：(Closed) 1 號櫃台：(Closed) 2 號櫃台： $5_1$ 、 $7_1$ 、 $2_1$ 、 $10_0$ 、 $6_0$ 、 $3_0$ 、 $4_2$ 、 $8_3$

由於假設櫃檯是環狀的，所以如果 0 號櫃台關閉會往 2 號櫃台開始找。

# Usage of testcase

在雲端中可以發現有測資，但可以發現這些測資是.in 檔與.out 檔，假設寫完 C++的程式如下

## Code

```
1. #include<iostream>
2.
3. using namespace std;
4.
5. int main(void)
6. {
7.     int a, b;
8.
9.     cin >> a >> b;
10.    cout << a + b << endl;
11.
12.    return 0;
13. }
```

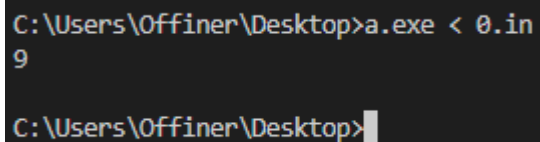
編譯後會產生一個 exe 的執行檔，而有個.in 的測資檔案，內容如下

0.in
4 5

假設產生的執行檔是 a.exe，而測資是 0.in，這時只要在終端機執行

a.exe < 0.in
--------------

就可以看到該執行檔輸出的結果



```
C:\Users\Offiner\Desktop>a.exe < 0.in
9
C:\Users\Offiner\Desktop>
```

再來就可以去對照 0.out 的檔案內容是否與執行結果相同!

# Submission Format

繳交格式請如下:

```
<your name>/
  q1/
    main.cpp
  q2/
    7000.csv
    main.cpp
  q3/
    pointer_table_screen_shot.jpg (png)
  q4/
    main.cpp
  q5/
    main.cpp
  q6/
    main.cpp
```

程式部分會透過 Shell Script 去跑測資，所以請照上述格式包好後上傳至繳交的雲端連結

可以的話，蠻建議大家可以用類別的形式來實現，因此有 func.cpp 與 func.h 也直接放入該問題資料夾中即可

Q5、Q6 是比較挑戰的題目，可以不用寫，由於比較需要一些資料結構上知識，有空的人可以挑戰看看~

這些問題是我突發奇想，測資也只是簡單產生，所以可能存在 Bug，因此對問題或測資有任何疑問、需要更多測資、想找我討論甚至是覺得題目太簡單沒挑戰性者，歡迎透過 Line 私訊我~