**程式設計(一)-HW03**

Due to 11/3 PM 11:59／授課老師：紀博文

**一、基本資料**

姓名：林育辰

系級：資工111

學號：40771131H

**二、檔案有哪些？**

1. hw0301.c
2. hw0302.c
3. hw0303.c
4. hw0304.c
5. hw0305.c
6. hw0306.c
7. Makefile
8. README.pdf

◎每個.c檔皆有詳細註解！問題的回答寫在README每題詳細說明中！

**三、如何執行？**

請輸入make→編譯hw0301.c~hw0306.c→產生hw0301~hw0306檔

指令如下：

$ make

$ ./hw0301

$ ./hw0302

…

以此類推，即可分別執行hw0301~hw0306

**四、索引**

**第一題**──────────────────P.2-4

第二題──────────────────**P.4-6**

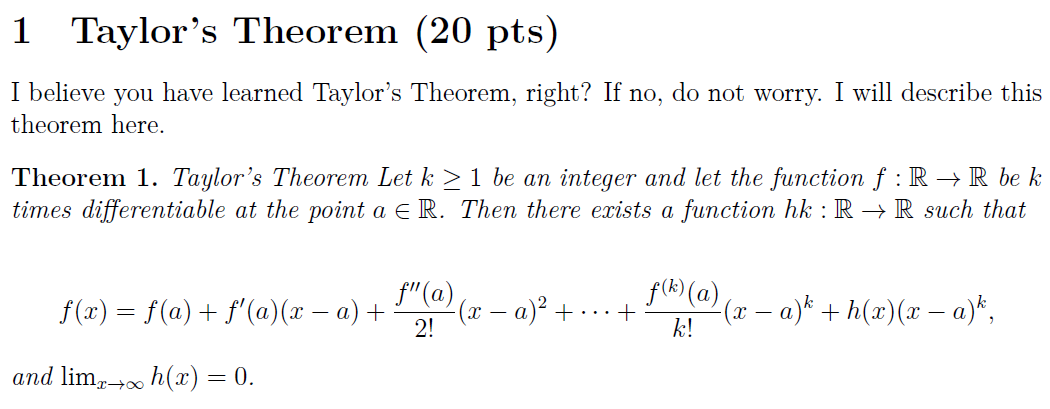
**第三題**──────────────────P.6-8

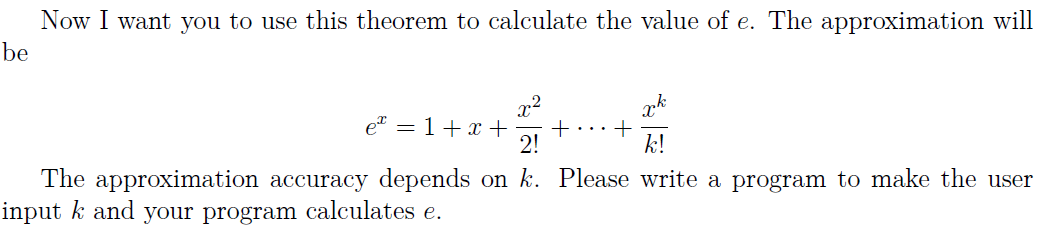
第四題──────────────────**P.9-10**

**第五題**──────────────────P.10-12

第六題──────────────────**P.12-14**

**說明**





**◎題意說明**→請使用泰勒展開式估算自然常數e的數字，e的估算型式如上，讓使用者輸入一非負整數，套進公式中，並輸入依據此k值所估算出的e值

**◎檢查**

1) k值必須為非負整數→若不符合則提示使用者”input error!”並重新輸入

**◎輸入格式**



編譯後，執行”$ ./hw0301”

並輸入一個非負整數(k)的值

**◎輸出格式**

1) 輸出依據k值計算出的自然常數數值

2) 此數值輸出到小數點後第二十位

3) 輸出第一行為以double型態計算的數值

4) 輸出第二行為以float型態計算的數值

**◎程式設計思路**

1) 以變數k儲存使用者輸入的k

→檢查！若數值<0則重新輸入

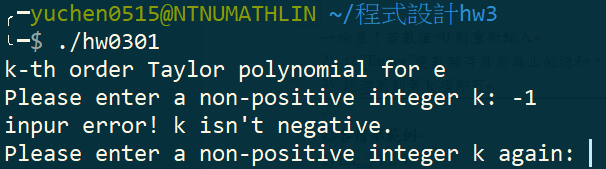
2) 以”Taylor”變數儲存目前為止的總和，以temp儲存每次乘以(1/i)後的數值

3) 跑迴圈到第k項即可

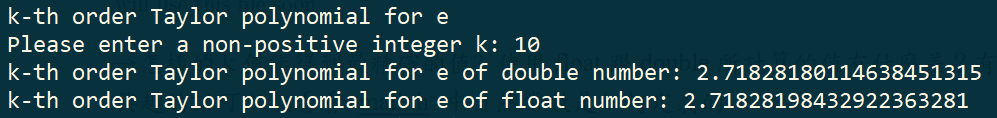
**◎各情形範例**

1) Ex: ”-1”

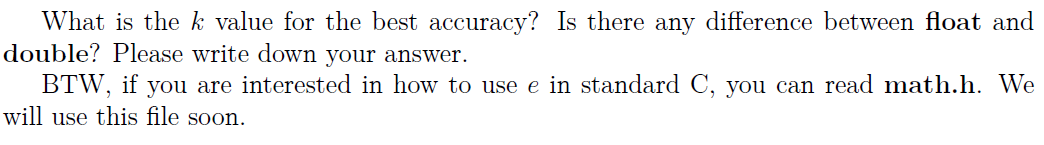
Tip: k值輸入為負，提示使用者重新輸入



2) Ex: ”10”

Tip: k值疊加到(x^10/10!)所估算的情形

**◎題目回應**



→怎樣的k值會得到最精確的值? 使用float跟double所計算的值有什麼差？有興趣的話可以翻看看<math.h>中自然常數是如何定義的

一般自然常數數字大略如下：



Ans:

1. 若以float計算k=9時，是最精確的

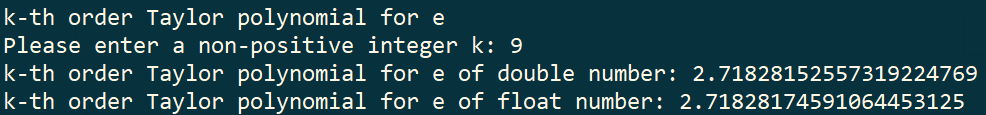
當k=6時，精確到2.718

k=7時，精確到2.7182

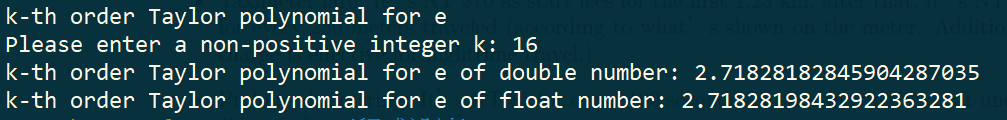
k=9時，精確到2.718281，估算為2.71828174591

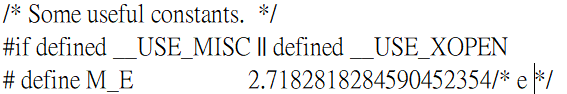
k=10時，精確到2.7182818 ，估算為2.718281984329

k=11時，精確到2.71828182



1. 而若以double計算時，則精確到小數點後第15位，而當k>=17時，數值是較精確的，如圖



1. 因此float和double，float有效數字是7位，到小數點後6位精確，之後因數值過小，超出float的精確度因此會出現k=9比k=10還精確的情形，超出他所能承受的精確度數值就不會是你原本要的數值而會有誤差，進一步影響到值，而double有效數字是16位，到小數點後15位精確
2. <math.h>中，對自然常數定義如下

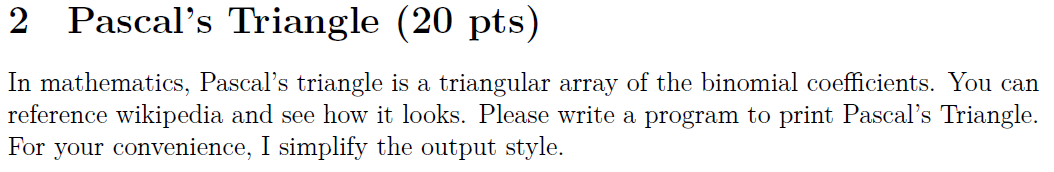
為20位有效數字的e，

而long double定義則如下



大致而言並不是經由計算，而是直接將計算好的數值定義為e值代入

**說明**

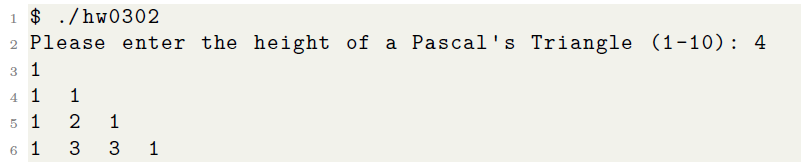


**◎題意說明**→使用者輸入帕斯卡三角形的高(h)，依照帕斯卡三角形的原則，寫出一個程式印出這種三角形

**◎檢查**

1) k值必須在區間[1,10]之內→若不符合則提示使用者”input error!”並重新輸入

**◎輸入格式**



編譯後，執行”$ ./hw0302”

並輸入一個介於閉區間[1,10](帕斯卡三角形)的高(integer)

**◎輸出格式**

1) 依列依行輸出印出帕斯卡三角形

**◎程式設計思路**

1) 儲存高度，並設定一個一維陣列儲存每行該印出的數值

2) 若使用二維陣列會很簡單好寫，但浪費記憶體又有些複雜

3) 若由左往右更改數值，容易將原本的數值蓋掉，又以新數值計算

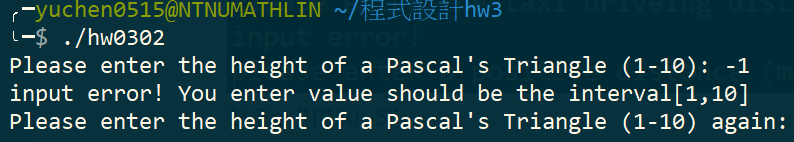
4) 由右往左掃，則不會有此問題，但要注意邊界上的處理

5) 第一行 1，在陣列中會不好處理，pascal三角形需要運用到[i] + [i-1]會超出邊界，因此先將此數值輸出(使用者輸入必介於[1,10]之間，故第一行必定會輸出

**◎各情形範例**

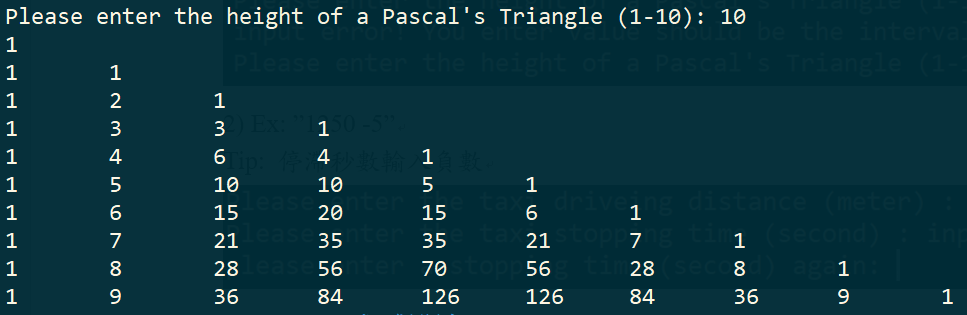
1) Ex: ”-1”

Tip: 高度(integer)輸入[1,10]以外的數值，提示使用者重新輸入



2) Ex: ”10”

Tip: 輸出高度為10的Pascal三角形



**◎題目回應**

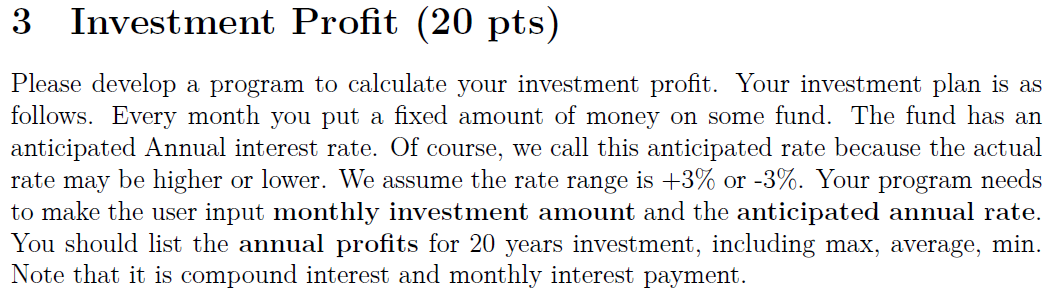


→你怎麼處理校正(排版)問題的呢？請分享你的方法

Ans:

1. 設計外迴圈、內迴圈，內輸出單行，外輸出多行，在內圈時輸出數值以**”\t”**定位方式排版，而內圈結束時以**”\n”**換行，即可整齊舒適
2. 主要是使用**陣列**處理，若不使用陣列時，可設置10個變數依序處理也可

**說明**



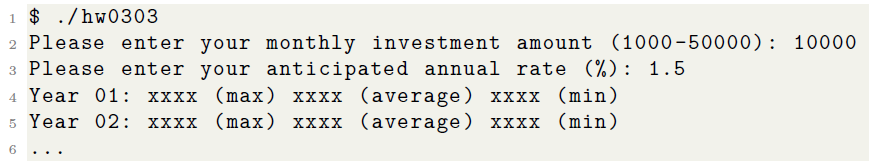
**◎題意說明**→請設計出一個程式，能讓使用者輸入每月存入的金額，**預期年利率**，依”月複利”方式計算，另題目設定**實際年利率**波動最大為”預期年利率+3%”，最小為”預期年利率-3%”去計算，請輸出每年複利和的max, average, min，其中max為以”**預期年利率+3%**”計算，aver以”**預期年利率**”計算，min以”**預期年利率-3%**”計算

※經詢問後，應**輸出複利和**，也就是包含本金、利息的金額

**◎檢查**

1) 檢查每月存入的金額是否介於[1000,50000]內，若否則提示使用者重新輸入

**◎輸入格式**



編譯後，執行”$ ./hw0303”

依提示輸入

1. 每月存入的額度(monthly investment amount)
2. 預期年利率(annual rate)

**◎輸出格式**

1) 依照順序，輸出 “Year xx: xxxx(max) xxxx(average) xxxx(min)”

**◎程式設計思路**

1) 應設置變數儲存使用者輸入的月額度、年利率，並須設置三個變數，依序存放最大可能、預期、最小可能的複利和為 max, aver, min

※另需特別檢查使用者輸入是否在[1000,50000]區間內

2) 計算實際最大利率、實際利率、實際最小利率的變數，並處理掉百分比，設計外迴圈跑20次，而內迴圈跑12次(一年有十二月)，其中內迴圈能夠計算該年經過十二個月的月複利以後的複利和，再藉此計算出該年最終複利和

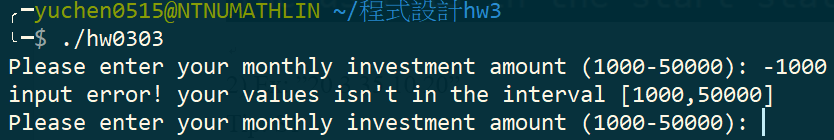
※不需扣除本金，我們需要輸出的是**複利和**

※預期年利率**不限多少百分比**(題目沒設限，只有每月金額有限)

**◎各情形範例**

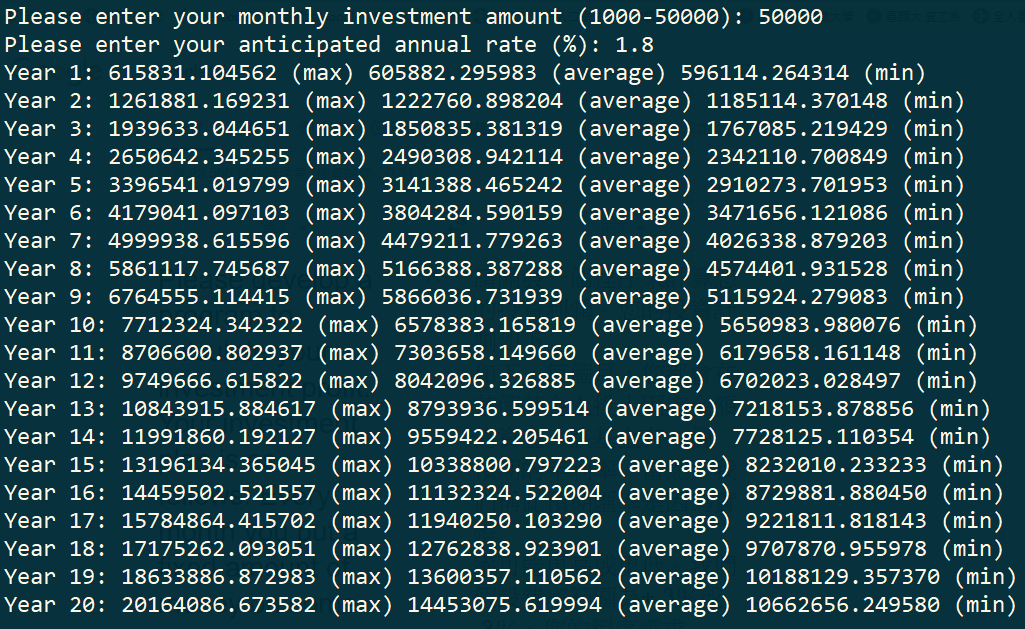
1) Ex: ”-1000”

Tip: 每月額度超出[1000,50000]的範圍



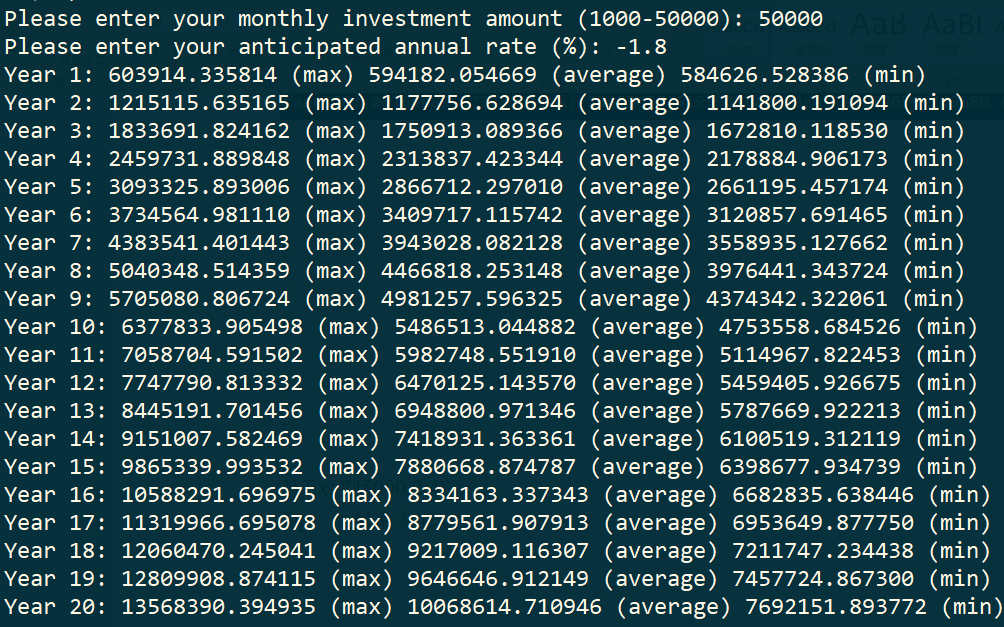
2) Ex: ”50000 1.8”

Tip: 年利率為正，實際利率最大為4.8%，最小為-1.2%，平均為預期利率1.8%

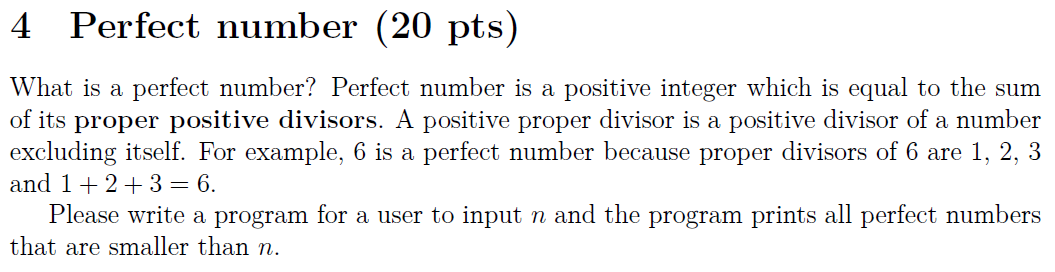


3) Ex: “50000 -1.8”

Tip: 年利率為負，實際利率最大為1.2%，最小為-4.8%，平均為預期利率-1.8%



**說明**



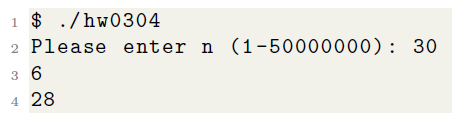
**◎題意說明**→使用者輸入一個n值介於[1,50000000]以內，我們需要設計出一個程式，找出從1到n(不包含)的所有完美數

→完美數定義：定一整數n，若其所有因數(不包含自己)相加後等於n時，則為perfect number

**◎檢查**

1) 使用者輸入的n值必須介於[1,50000000]以內，若否則提示使用者重新輸入

**◎輸入格式**



編譯後，執行”$ ./hw0304”

依提示輸入一個介於1~50000000的整數n

**◎輸出格式**

1) 輸出1~n-1的所有完美數

※注意：詢問助教該題時間限制後，回覆為本題**無時間限制**

**◎程式設計思路**

1) 儲存n值，並設置sum\_divisor處理因數總和的問題

2) 外迴圈由1跑到n-1，內迴圈則由1到(int)sqrt(i)+1檢查因數，若是則疊加

3) 到(int)sqrt(i)+1有幾個考量： 若i等於9，則檢查到3

若i等於8，則原本sqrt(i)應是2.XXX，取int為2，再+1=3，

此時3\*3已經大於8了，故按照這樣的方式可以檢查到所有因數

4) EX:令i=10，則因式有 1\*10 = 10／2\*5=10／5\*2=10／10\*1=10

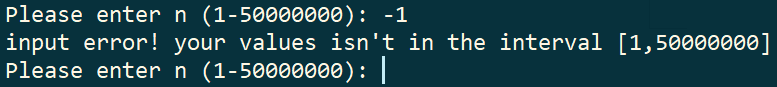
我們將只需要跑到3，就能檢查到1\*10, 2\*5，取出1和2的同時，再取出10,5，即可不用算到後面去

5) 若迴圈中因數總和早已超出原本的數則跳出迴圈

**◎各情形範例**

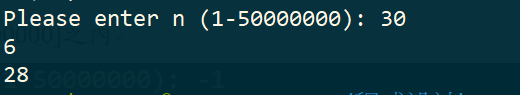
1) Ex: ”-1”

Tip: n值不在[1,50000000]之內



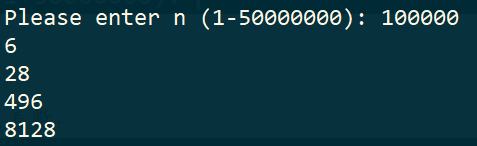
2) Ex: ”30”

Tip: 取出1-30間的完美數

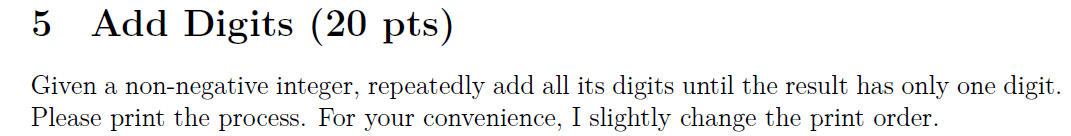


3) Ex: ”100000”

Tip: 取出1-100000間的完美數



**說明**

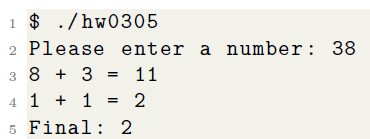


**◎題意說明**→使用者輸入一個非負的整數 (小於2^31 -1)，需要寫一個程式將她每一位數相加，並且再以相加後的數字重複數次動作，直到僅剩一位數為止，輸出”Final: (該數值)”，其中過程的計算必須輸出

**◎檢查**

1) 使用者輸入是否為非負整數，若否則提示使用者重新輸入

**◎輸入格式**



編譯後，執行”$ ./hw0305”

依提示輸入一個非負整數n

**◎輸出格式**

1) 將各位數疊加的過程輸出，並不斷疊加，直到位數僅剩一位，再輸出最終結果

**◎程式設計思路**

1) 以num變數儲存使用者輸入，而以temp變數放入num，由此數作為檢查的條件，另以sum儲存每次位數疊加的結果，在最後將此數賦予temp

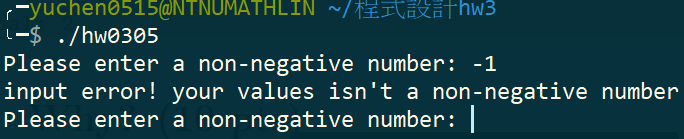
2) 於是外圈設置迴圈條件(temp>=10)，內圈則在跑一個while(temp>0)的迴圈，在其內則判斷temp>=10以及10>temp>=0，以便”+”號,”=”號的輸出

3) 跳出迴圈後，此時temp留存的數值即為最終結果，將此輸出即可

**◎各情形範例**

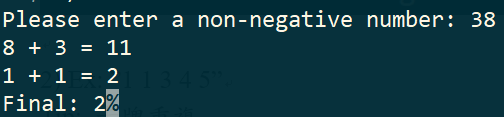
1) Ex: ”-1”

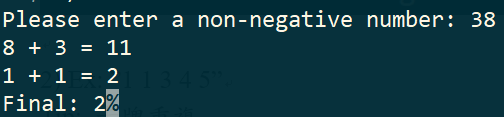
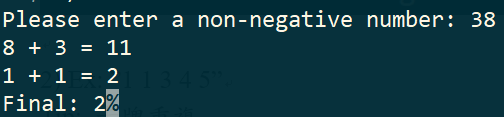
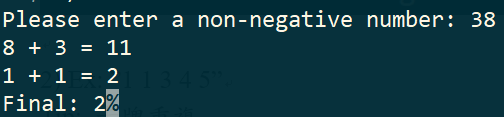
Tip: 輸入負數



2) Ex: “38”

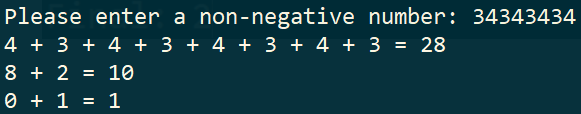
Tip: 僅有一個”+”、一個”=”的情形



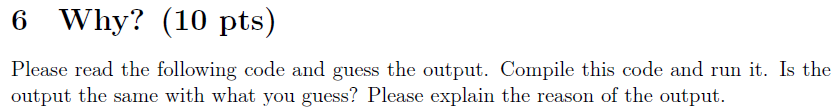


3) Ex: “34343434”

Tip: 第一次digit有多個加號的情形



**說明**



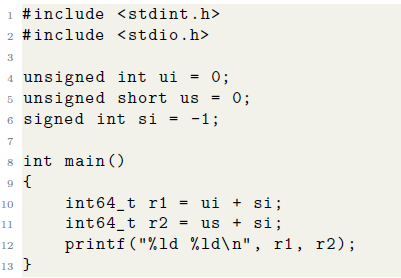
**◎題意說明**→

(1)閱讀以下程式碼，並猜測他的結果

(2)實際編譯並執行這隻程式碼

(3)這結果是否跟你的猜測相同

(4)請解釋為何出現這樣數值的原因

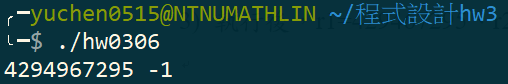


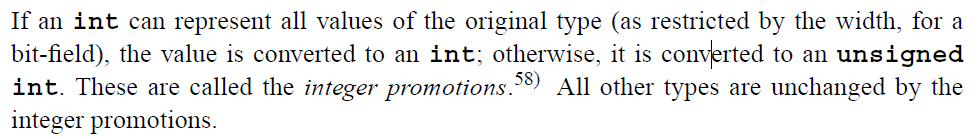
**◎回答**→

1) 由於ui值的範圍從0一直到2^32，而si值則從-2^31~2^31-1，兩者範圍大小是一致的，猜測unsigned int和signed int，雖範圍一樣但相加時應以unsigned型態計算，ui+si = -1，但因為是unsigned int型態此時往下溢位，則跑到最大值-1，因此**猜測數值應為2^32-1**

2) us的值為unsigned short int，其中範圍從0到2^16-1，si則為signed int，範圍則由-2^31~2^31-1，兩者因si範圍較大，猜測運算時應會併入範圍較大的signed int，故us+ si = (signed int) -1，再和變數r2(int64\_t)合併，變為**(int64\_t) -1**

3) 執行後，r1=429467295、r2=-1，與**猜測完全相符**



4) 

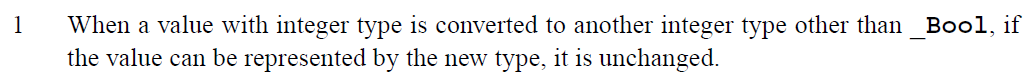
根據C11文件所述，當int囊括了原始型態的所有範圍，則將此數值轉為int計算，倘若無法涵蓋原始型態的範圍，則轉為unsigned int計算。

→當我們將ui(unsigned int)+si(signed int)時，

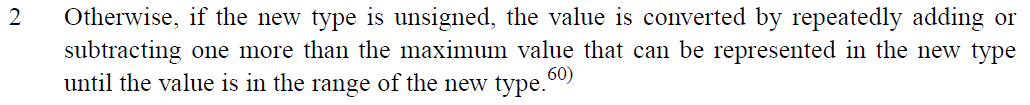
1. signed int範圍從-2^31到2^31-1
2. signed int範圍從0到2^32-1

→int無法涵蓋2^31~2^32-1的範圍

→計算時以ui(unsigned int) + si(unsigned int)處理



上圖說明道「若由A的整數型態轉變到B的整數型態，如果值能被B的整數型態表示時，其值不會更動」



而如果新型態是unsigned，數值不在範圍中，則重複加或減，直到該值在unsigned的範圍中(處理溢位)

故ui(unsigned int) + si(signed int) =>

ui(unsigned int) + si(unsinged int) => -1(unsigned int)→

範圍的最大值減1 => 2^32-1

另外一個輸出則為 us(unsigned short) + si(signed int)

→unsigned short在[0,2^16-1]範圍內

→signed int在[-2^31,2^31-1]範圍中

→int可完全涵蓋unsigned short的值→us(unsigned short) + si(signed int)

→us(signed int) + si(signed int) = -1(signed int)=>-1(int64\_t)