程式設計(一)-HW06

Due to 01/05 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) hw0601.c
2) hw0601.h
3) hw0601_header.c
4) hw0602.c
5) hw0603.c
6) hw0604.c
7) hw0604test.h
8) hw0605.c
9) hw0605test.h
10) hw0606.c
11) Makefile
12) 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！
◎ hw0601_header.c 和 hw0601.h 為 原先助教提供的 **hw0601.h** 分開寫，在.h 檔內單純做 function prototype，而在 hw0601_header.c 中則詳細寫入函式所要做的事，**因此若助教需要更改測資，請更改 hw0601_header.c**，有原先助教檔案的痕跡~

三、如何執行？

請輸入 make→編譯 hw0601.c~hw0606.c→產生 hw0601~hw0606 檔
指令如下：

```
$ make
$ ./hw0601
$ ./hw0602
…
```

以此類推，即可分別執行 hw0601~hw0606

四、索引

第一題——P.2-6
第二題——**P.6-10**
第三題——P.11-13
第四題——**P.14-16**
第五題——P.16-18
第六題——**P.18-23**

說明

1 Poker Again (20 pts)

Let's play poker!! In some poker games, the player sorts his hand card by the card number instead of the card suit. The card encoding is as follows:

- 1-13: ♠ Ace to King.
- 14-26: ♡ Ace to King.
- 27-39: ♢ Ace to King.
- 40-52: ♣ Ace to King.

You should implement a shuffle function and output four hand-cards for four players. Then you should sort their cards in the ascending order according to the card value, not suit. That is, you must implement the following functions:

```
1 void print_card( const uint8_t player[13] );
2 int32_t sort_card( uint8_t player[13], int32_t ( * compare)( int32_t a,
    int32_t b ) );
```

There are three sorting ways defined as follows.

1. According to the card value in the ascending order.
2. According to the card value in the ascending order, but "2" is the biggest number.
3. According to the card suit, ♠ → ♡ → ♢ → ♣. Each suit should be sorted in the ascending order according to the card value.

Note that these function should be placed in another file different with the main function. The main function should be as follows:

shuffle will be provided by TA in **hw0601.h**. You must include the header file and call **shuffle** in your main function.

Of course, if TA gives you an unreasonable hand of cards, you need to print a warning message.

◎題意說明→ 寫出程式，能根據 shuffle function 的數值，按照 type1, type2, type3 的說明排序，並分別輸出，你必須定義好四個 players 陣列去存玩家的手排(至於會拿到什麼手牌，是根據 shuffle 寫死的 int card 去分配)，並且使用 pointer function 實作，以下為三種類型的排序：

1) 按卡牌的數值由小到大排序(不需排花色)
2) 按卡牌的數值由小到大排序，但 2 為最大，由小至大順序應為 3456789 10 JQK12
3) 先按卡牌的數值由小到大排序，點數一樣則按黑桃→愛心→方塊→梅花順序排

→最終依序輸出 3 種排序的結果

※注意

1) 必須檢查助教輸入的卡牌是否有「作弊」，即一個花色一個數值只會出現一張，而且張數總數應為 52 張
2) 關於第三種類型的排序方式，已詢問多個同學的解讀方式，一半認為是「先排序數字，再排序花色。」另一半認為是「先排序花色，再排序數字。」寄信詢問助教後似乎是**前者**，但為了避免這樣模稜兩可的情況，在本程式**兩種結果都會輸出**，即 **type3 為「先排序數字，再排序花色」，type4 為「先排序花色，再排序數字」**以免爭議。(不過如果是先排花色，再排數字，直接按照數字由小排到大就好了，根本沒必要出題，因此個人想法是「先排數字再排花色」。

◎輸入格式

```
uint8_t player1[13] = {0};
uint8_t player2[13] = {0};
uint8_t player3[13] = {0};
uint8_t player4[13] = {0};
shuffle( player1, player2, player3, player4 );
printf( "Before:\n" );
print_card( player1 );
print_card( player2 );
print_card( player3 );
print_card( player4 );
printf( "Type 01:\n" );
sort_card( player1, func01 );
sort_card( player2, func01 );
sort_card( player3, func01 );
sort_card( player4, func01 );
print_card( player1 );
print_card( player2 );
print_card( player3 );
print_card( player4 );
printf( "Type 02:\n" );
sort_card( player1, func02 );
sort_card( player2, func02 );
sort_card( player3, func02 );
```

```
sort_card( player4, func02 );
print_card( player1 );
print_card( player2 );
print_card( player3 );
print_card( player4 );
printf( "Type 03:\n" );
sort_card( player1, func03 );
sort_card( player2, func03 );
sort_card( player3, func03 );
sort_card( player4, func03 );
print_card( player1 );
print_card( player2 );
print_card( player3 );
print_card( player4 );
```

↓額外添加(為了防止解讀不同的另個情況)

```
//四個玩家手牌「先排花色，再排數字。」
printf("Type 04:\n");
sort_card(player1, func04);
sort_card(player2, func04);
sort_card(player3, func04);
sort_card(player4, func04);
print_card(player1);
print_card(player2);
print_card(player3);
print_card(player4);
```

編譯後，執行"$ ./hw0601"

→若手牌沒有出老千，則依序按三種排序方式輸出，其中第三種個人解讀為「先排數字，再排花色」，為防解讀不同，會輸出第四種「先排花色，再排數字」。

→若老牌有出老千，則輸出如下：

```
//有人出老千就提示Warning Messege
printf("The following are possible conditions\n");
printf("1) More than four cards with one point.\n");
printf("2) Range isn't between [1,52]\n");
printf("Warning Messege!(someone cheating)\n");
```

◎檢查

1) 有沒有[1,52]以外的牌，有→輸出錯誤
2) 有沒有重複的牌，有→輸出錯誤 (如兩張梅花 1，是不可能的)

◎輸出格式

```
┌─yuchen0515@NTNUMATHLIN ~/程式設計hw6
└─$ ./hw0601
Before:
45 26 10 43 19 51 50 16 36 31 20 38 41
37 34 7 13 12 27 5 48 46 42 1 6 21
17 28 8 15 2 33 4 3 22 29 9 11 49
40 30 35 24 39 44 23 52 14 47 32 18 25
Type 01:
41 16 43 31 45 19 20 10 36 50 51 38 26
27 1 42 5 6 7 46 34 21 48 37 12 13
28 15 2 3 29 17 4 33 8 22 9 49 11
40 14 30 44 18 32 47 35 23 24 25 39 52
Type 02:
16 43 31 45 19 20 10 36 50 51 38 26 41
42 5 6 7 46 34 21 48 37 12 13 27 1
3 29 17 4 33 8 22 9 49 11 28 15 2
30 44 18 32 47 35 23 24 25 39 52 40 14
Type 03:
41 16 43 31 19 45 20 10 36 50 38 51 26
1 27 42 5 6 7 46 21 34 48 37 12 13
2 15 28 3 29 4 17 33 8 9 22 49 11
14 40 30 18 44 32 47 35 23 24 25 39 52
Type 04:
10 16 19 20 26 31 36 38 41 43 45 50 51
1 5 6 7 12 13 21 27 34 37 42 46 48
2 3 4 8 9 11 15 17 22 28 29 33 49
14 18 23 24 25 30 32 35 39 40 44 47 52
```

◎程式設計思路

1) 沒什麼特別，純粹考會不會 pointer function，就連 shuffle, card 助教都幫忙寫好了，只要會寫 compare function 即可

2) 在排序時，做了特別的處理，如由小排到大，因 1,14,27,40 應為最小，要利用餘數去判斷數字時，寫成(a+12)%13，可以將最小的數值 1，透過這式子變 0，而最大的 K，則為(25)%13=12，為最大數值。

3) 若 2 為最大，則 3 是最小，因此寫成(a+10)%13 就能解決。

4) 寫一個 check_card 的 function，檢查牌是否超出範圍，或重複，排序或輸出時就要直接檢查。

說明

2 Finite State Machine v2 (20 pts)

You still remember this problem, right? Figure 1 is your homework. However, there is something different here. First, all inputs are from 0-255 except the stop signal -1. That is, the user can input as many integers as he wishes and he should use -1 as the last input. Second, you should not use **switch**. Instead, you **must** use an array of function pointers.

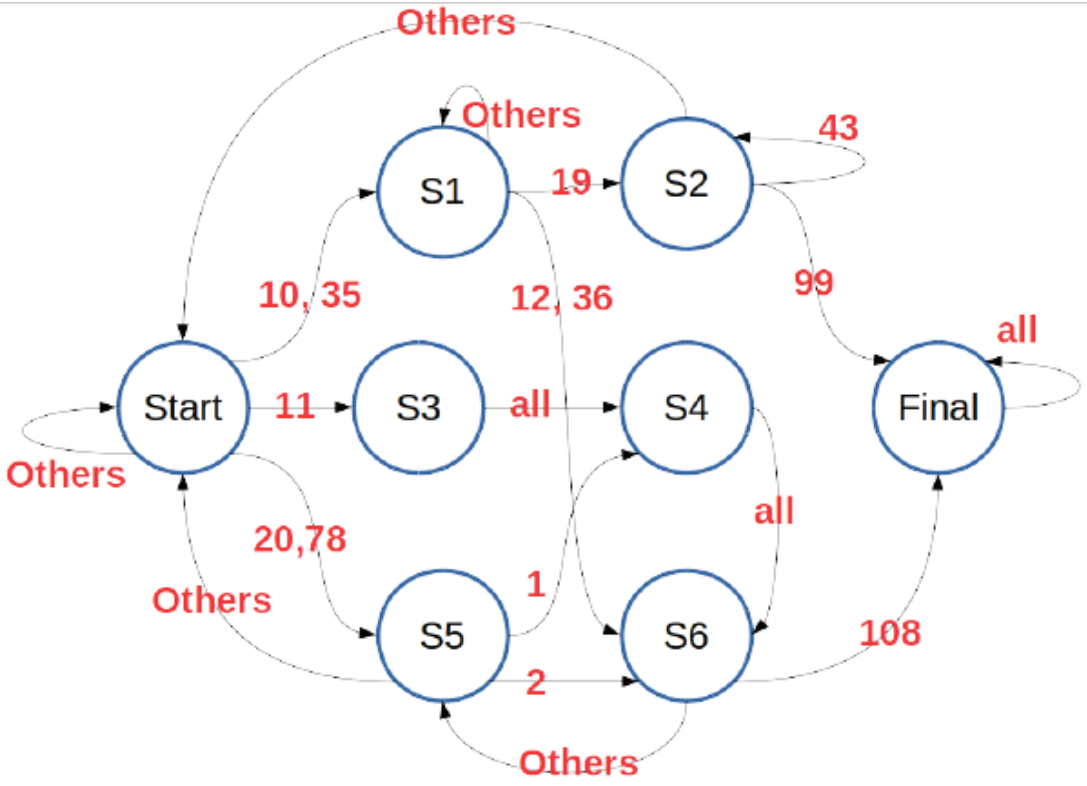


Figure 1: Deterministic Finite Automata.

◎題意說明→你還記得這題目吧？是之前你的作業(自動機)，不過現在有些不同。請讓使用者輸入[0,255]內的數，若輸入-1 則代表中斷。不要使用 switch，你必須使用 array of pointer functions 來作答，給一點提示，參考如下：

I will give you a hint. You can write code like this:

```
1  void state_start(int *state, int input);
2  void state_s01  (int *state, int input);
3  ...
4
5  int main()
6  {
7      int (*dfa[8])( int *, int );
8
9      dfa[0] = state_start;
10     dfa[1] = state_s01;
11     ...
12
13 }
```

※注意：上圖**第七行應改為 void (*dfa[8])(int * , int);**

◎檢查

1) 數字必須位於[0,255]內，或為-1(中斷)，否則提示使用者重新輸入。

◎輸入&輸出格式

```
$ ./hw0602
Please enter an integer (0-255, -1: stop): 35
Please enter an integer (0-255, -1: stop): 19
Please enter an integer (0-255, -1: stop): 43
Please enter an integer (0-255, -1: stop): 99
Please enter an integer (0-255, -1: stop): 10
Please enter an integer (0-255, -1: stop): -1
The user is in the final state.
```

編譯後，執行"$ ./hw0602"，輸入：

1) 不斷輸入[0,255]間的整數，直到你輸入-1 則停止
2) 會輸出「The user is in the _____ state.」

◎程式設計思路

1) 如提示所寫，宣告 void (*dfa[8])，並依序存入 state_start, state_s1, …, state_final 八個函式，和一個狀態初始化為 0(剛開始 state 在 start 的位置)
2) 當我輸入 dfa[state]就能跳到相對應的函式，處理目前的 state 和 input，如"dfa[state](&state, input);"
3) 跑無限迴圈，若輸入-1 則 break，輸入[0,255]以外的數則 continue 讓使用者重新輸入，正確輸入則直接跳到相對應的函式，只需更改 state 的數值，而各函式內只需根據自動機的圖，按照條件去更改 state 即可(state 為 0~7 的數值，剛好和 dfa 的八個數值相對應)
4) 創造一個字串陣列，存放 start, s1, s2,…, final，最後直接 printf("The user is in the %s state.\n", str[state]);就能輸出相對應的句子(狀態)

◎各情形範例

1) Ex: ”35 19 43 99 10 -1”

Tip: test case #1

```
┌─yuchen0515@NTNUMATHLIN ~/程式設計hw6
└─$ ./hw0602
Please enter an integer (0-255, -1: stop): 35
Please enter an integer (0-255, -1: stop): 19
Please enter an integer (0-255, -1: stop): 43
Please enter an integer (0-255, -1: stop): 99
Please enter an integer (0-255, -1: stop): 10
Please enter an integer (0-255, -1: stop): -1
The user is in the final state.
```

2) Ex: ”10 19 43 45 80 -1”

Tip: start

```
Please enter an integer (0-255, -1: stop): 10
Please enter an integer (0-255, -1: stop): 19
Please enter an integer (0-255, -1: stop): 43
Please enter an integer (0-255, -1: stop): 45
Please enter an integer (0-255, -1: stop): 80
Please enter an integer (0-255, -1: stop): -1
The user is in the start state.
```

3) Ex: ”11 23 43 50 50 35 -1”

Tip: s1

```
Please enter an integer (0-255, -1: stop): 11
Please enter an integer (0-255, -1: stop): 23
Please enter an integer (0-255, -1: stop): 43
Please enter an integer (0-255, -1: stop): 50
Please enter an integer (0-255, -1: stop): 50
Please enter an integer (0-255, -1: stop): 35
Please enter an integer (0-255, -1: stop): -1
The user is in the s1 state.
```

4) Ex: ”11 50 50 50 50 10 19 43 -1”

Tip: s2

```
Please enter an integer (0-255, -1: stop): 11
Please enter an integer (0-255, -1: stop): 50
Please enter an integer (0-255, -1: stop): 50
Please enter an integer (0-255, -1: stop): 50
Please enter an integer (0-255, -1: stop): 50
Please enter an integer (0-255, -1: stop): 10
Please enter an integer (0-255, -1: stop): 19
Please enter an integer (0-255, -1: stop): 43
Please enter an integer (0-255, -1: stop): -1
The user is in the s2 state.
```

5) Ex: ”11 -1”

Tip: s3

```
Please enter an integer (0-255, -1: stop): 11
Please enter an integer (0-255, -1: stop): -1
The user is in the s3 state.
```

6) Ex: ”50 11 50 -1”

Tip: s4

```
Please enter an integer (0-255, -1: stop): 50
Please enter an integer (0-255, -1: stop): 11
Please enter an integer (0-255, -1: stop): 50
Please enter an integer (0-255, -1: stop): -1
The user is in the s4 state.
```

7) Ex: ”20 50 78 -1”

Tip: s5

```
Please enter an integer (0-255, -1: stop): 20
Please enter an integer (0-255, -1: stop): 50
Please enter an integer (0-255, -1: stop): 78
Please enter an integer (0-255, -1: stop): -1
The user is in the s5 state.
```

8) Ex: ”78 1 50 -1”

Tip: s6

```
Please enter an integer (0-255, -1: stop): 78
Please enter an integer (0-255, -1: stop): 1
Please enter an integer (0-255, -1: stop): 50
Please enter an integer (0-255, -1: stop): -1
The user is in the s6 state.
```

9) Ex: ”11 50 60 80 1 20 70 90 10 50 19 43 90 90 11 50 50 108 -1”

Tip: final

```
Please enter an integer (0-255, -1: stop): 11
Please enter an integer (0-255, -1: stop): 50
Please enter an integer (0-255, -1: stop): 60
Please enter an integer (0-255, -1: stop): 80
Please enter an integer (0-255, -1: stop): 1
Please enter an integer (0-255, -1: stop): 20
Please enter an integer (0-255, -1: stop): 70
Please enter an integer (0-255, -1: stop): 90
Please enter an integer (0-255, -1: stop): 10
Please enter an integer (0-255, -1: stop): 50
Please enter an integer (0-255, -1: stop): 19
Please enter an integer (0-255, -1: stop): 43
Please enter an integer (0-255, -1: stop): 90
Please enter an integer (0-255, -1: stop): 90
Please enter an integer (0-255, -1: stop): 11
Please enter an integer (0-255, -1: stop): 50
Please enter an integer (0-255, -1: stop): 50
Please enter an integer (0-255, -1: stop): 108
Please enter an integer (0-255, -1: stop): -1
The user is in the final state.
```

說明

3 Rotation (20 pts)

Given a 2-d coordinate and θ, please rotate this point with θ clockwise where the circle center is O, as shown in figure 2.

◎題意說明→給一個二維座標(x, y)點 P 和 theta 角度，請輸出 P 點**順時針**旋轉 theta 度後的(x',y')數值為何(示意圖如下)

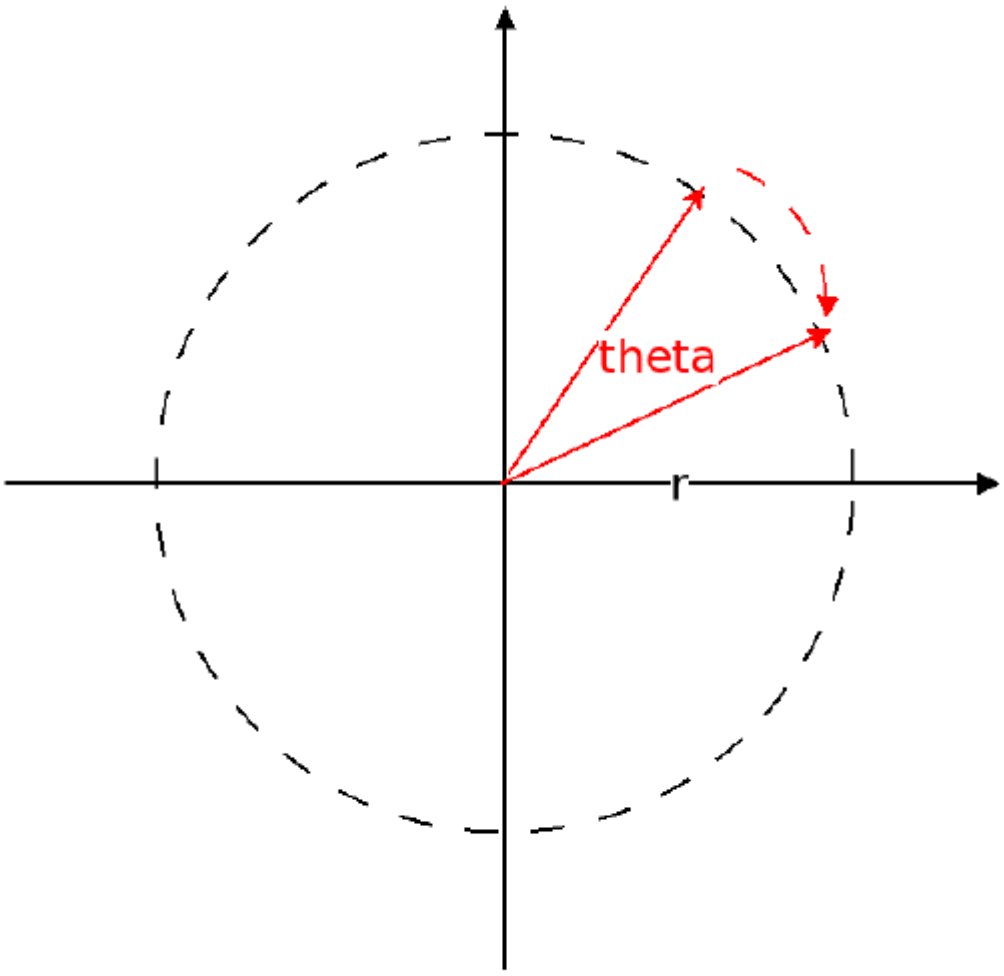


Figure 2: Rotation.

Note that you need to implement the following function. The coordinate after rotation should be returned in the input arguments.

```
1 void rotate( double *x, double *y, double theta );
```

※注意

1) 應以如上圖的 function 方式操作

→void rotate(double *x, double *y, double theta);

2) 使用者輸入角度應在[0,360]範圍內

3) Math.h 的三角函數，其 Input 應為”弧度”而非一般角度

4) 座標、角度皆可為浮點數

5) 該程式輸出到小數點後第三位(teacher tell us that “**precision isn’t the concern in this class**.”

◎輸入&輸出格式

The behavior of your program should be as follows.

```
$ ./hw0603
Please enter a point: 0 1

Please enter theta (0-360): 90
1 0
```

Again, precision is not the concern in this class.

編譯後，執行”$ ./hw0603”

1) 輸入座標 x, y

2) 輸入 theta 角

3) 輸出座標 x’,y’

◎程式設計思路

1) 因為要輸入的是弧度，先定義出 pi 的數值，這裡利用 Machin 法即：

$$pi = 16\tan^{-1}(1/5) - 4\tan^{-1}(1/239)$$

2) 利用旋轉矩陣公式

$$M(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

3) 在數學上，旋轉矩陣以逆時針為正，題目要的是逆時針，因此上圖的 theta 用-theta 取代即可

4) 利用 弧度= 角度* (pi/180)，將 theta 轉弧度後，直接套旋轉矩陣，即可得出答案

◎各情形範例

1) Ex: ”0 1／90”

Tip: test case #1

```
┌─yuchen0515@NTNUMATHLIN ~/程式設計hw6
└─$ ./hw0603
Please enter a point: 0 1
Please enter theta (0-360): 90
1.000 0.000
```

2) Ex: ”1 0／90”

Tip: e_1 (standard vector)順時針轉 90 度

```
Please enter a point: 1 0
Please enter theta (0-360): 90
0.000 -1.000
```

3) Ex: ”0 1／180”

Tip: e_2 轉 180 度

```
Please enter a point: 0 1
Please enter theta (0-360): 180
0.000 -1.000
```

4) Ex: “1 0／180”

Tip: e_1 轉 180 度

```
Please enter a point: 1 0
Please enter theta (0-360): 180
-1.000 0.000
```

說明

4 Big Integer Multiplication (20 pts)

Let's implement big integer multiplication. We use an **uint8_t** array to present a big number. For example, 12345 can be presented as figure 3.

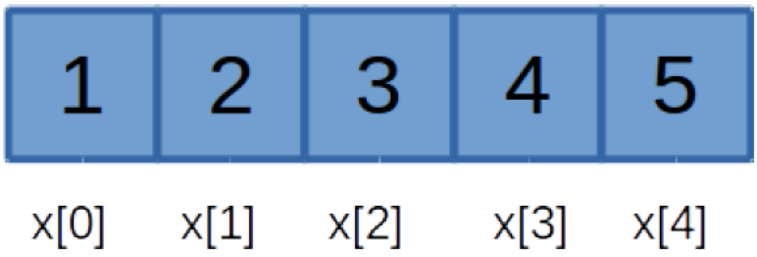


Figure 3: Big Number.

◎題意說明→請實作「大數乘法」，並使用 uint8_t 去存放每個數字，如圖"12345"，分別存進 x[0]~x[4]內

Now, please implement the following function. The output should be placed on **result**. Of course, you should allocate enough memory of result.

```
void multiplication( uint8_t **result, uint8_t *size,
                     uint8_t *x, uint8_t size_x,
                     uint8_t *y, uint8_t size_y );
```

而你應該實作的 function 如上，結果應存放在 result 內，當然你應該 memory allocate 足夠的記憶體空間給 result 使用

For your simplicity, you should include hw0604test.h. This header file will contain two arrays and their size. Of course, TA will change hw0604test.h when evaluating your code.

The behavior of your program should be as follows. In this example, these two numbers are 11 and 11.

為了方便起見，記得 include hw0604test.h，這份檔案包含兩個陣列和大小，然而 TA 會更改 hw0604test.h 的資料用來檢驗你的 code 輸出是否正確。

◎注意

1) **更改測資，請更改 hw0604test.h**
2) 陣列有多少數字，大小就應該有相對應的數字。

◎輸出格式

```
$ ./hw0604
121
```

編譯後，執行"$ ./hw0604"
實作 11*11，結果應該如上

◎程式設計思路

1) 模擬長乘法，但在 hw0604test.h 中，例如 x[4]是定義為{2,0,1,9}，也就是說最高位數的位置是 x[0]，不便於計算，因此寫一個 reverse 的函數將 x 存成{9,1,0,2}，從 9 往後累加計算會比較方便

2) 如測資已經翻轉後應為 x=>{9,1,0,2},

y=>{3,5,4,3,2,5,7,6,5,4,1,3,2,1,7,1,2,1}

我們的長乘法，應是 x[0]乘上 y 的每一項，而存放位置應該為 i+j(如 9*3,應存在 0+0 位，也就是第一位，如 x[0]*y[1]=9*5,存放在 0+1=1 位上

3) 例如今天 x[0]*y[1]=9*5，存放在第一位上(result[1])，所以在該格應存放 45，但因為>=10，因此將 45/10➔4，將 result[2]+=4 即可完成進位，但因為在 result[1]還是 45，因此 result[1]%10=5 即可完成進位。

4) 由於 uint8_t 只能存到 256，而最大位應是 9*9=81，但如果一次乘完，再一次進位，有可能會溢位！因此每次乘完就馬上進位，就能解決這問題。

5) 最終存放的結果為 result= {7,0,6,1,5,8,9,2,8,1,1,1,3,6,1,7,3,7,5,4,2}，而要輸出結果應該從最後面的 2 往前印，跑迴圈一個一個印完再換行即可，不過如果最大位放的是 0，則要記得 skip，直到第一個非零的數字出現後才開始一律輸出，如 00234，我應該要輸出 234，所以遇到第一個非零數字 2 才開始一律輸出，也就是輸出 2,3,4。

◎各情形範例

```
uint8_t x[4] = {2, 0, 1, 9};
uint8_t size_x = 4;
uint8_t y[18] = {1, 2, 1, 7, 1, 2, 3, 1, 4, 5, 6, 7, 5, 2, 3, 4 ,5, 3};
uint8_t size_y = 18;
```

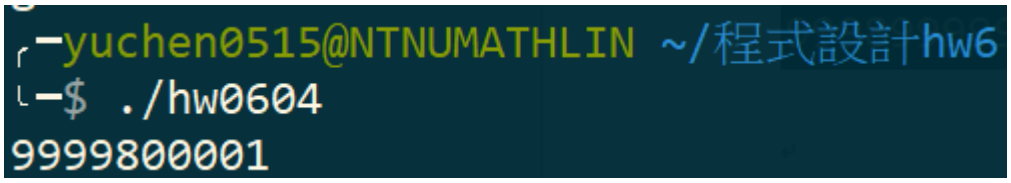
測資 1 助教範例測資

```
┌─yuchen0515@NTNUMATHLIN ~/程式設計hw6
└─$ ./hw0604
245737163111829851607
```

測資 1 輸出

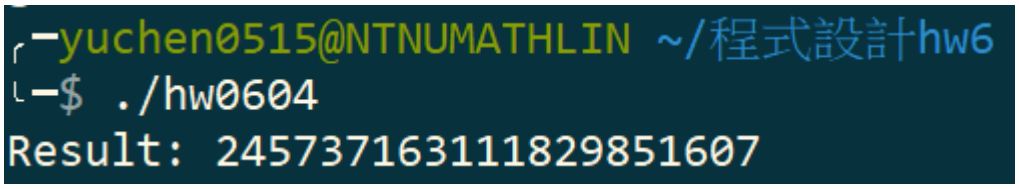
```
uint8_t x[5] = {9, 9, 9, 9, 9};
uint8_t size_x = 5;
uint8_t y[5] = {9, 9, 9, 9, 9};
uint8_t size_y = 5;
```

測資 2 容易溢位的測資



測資 2 輸出

※注意：因很早就寫好了，是採直接輸出，之後覺得輸出 Result: _____的形式比較漂亮，如圖：



測資 1 輸出(改變輸出型式為 Result: _____)

說明

5 Big Integer Division (20 pts)

Just like the last problem, however, this time you need to implement the division operation.

```
// x / y
void division( uint8_t **quotient, uint8_t *quotient_size,
               uint8_t **remainder, uint8_t *remainder_size,
               uint8_t *x, uint8_t size_x,
               uint8_t *y, uint8_t size_y );
```

For your simplicity, you should include hw0605test.h. This header file will contain two arrays and their size. Of course, TA will change hw0605test.h when evaluating your code.

◎題意說明→如同第四題，今天要實作的是「大數除法」，請寫一隻副函式，如上圖，而測試資料會放在 hw0605test.h，所以請記得 include 他，同樣的，包含兩個陣列和其大小，助教會修改 hw0605test.h 來去測試你的程式輸出結果有沒有問題。

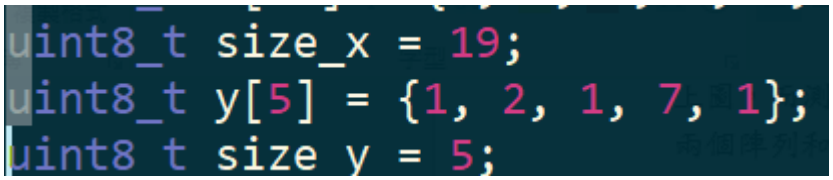
◎輸入&輸出方式

編譯後，執行"$ ./hw0605"

→最終輸出「商數」，和「餘數」

◎注意

1) **更改測資請更改 hw0605test.h**
2) 測資的陣列大小，要和變數的 size 相符合 Ex: {1,2,3,4,5}就要給 size=5
3) 因 test.h 內的 size_x, size_y 都是 uint8_t，已知輸入的兩條大數，位數不會超過 255，既測資為此種型態，表示不用將長度用大數的方式存，直接寫即可



◎程式設計思路

1) 想法是模擬長除法，我們長除法，都是看除數有幾位數，被除數有幾位數，就知道商應該從被除數最大位往後移除數的位數開始試，從 9~0 試試看，哪個數字最近，卻不超過原本數字就用那個

2) 要實現這個想法，應該實作出"大數的常數乘法"，"大數減法"，"大數的比大小"，原本覺得陣列應該常常要初始化，也有寫一個初始化的函數，之後因為用 calloc 所以沒用到

3) 商數從 9 往 0 試，依序乘上除數，並測驗原本那一條數字有沒有大於他，大於等於的話代表他是最適當的商數，就開始減，並把減出來的數字乘以 10，並從原本 x 那條移入下一個位數，繼續往下算
→如此即可計算大數除法

◎各情形範例

```
uint8_t x[19] = {2, 0, 1, 9, 1, 7, 6, 1, 3, 8, 2, 3, 5, 1, 4, 5, 6, 7, 1};
uint8_t size_x = 19;
uint8_t y[5] = {1, 2, 1, 7, 1};
uint8_t size_y = 5;
```

測資 1 助教範例測資

```
┌─yuchen0515@NTNUMATHLIN ~/程式設計hw6
└─$ ./hw0605
Quotient: 165900594711621
Remainder: 6480
```

測資 1 輸出

```
uint8_t y[19] = {2, 0, 1, 9, 1, 7, 6, 1, 3, 8, 2, 3, 5, 1, 4, 5, 6, 7, 1};
uint8_t size_y = 19;
uint8_t x[5] = {1, 2, 1, 7, 1};
uint8_t size_x = 5;
```

測資 2 助教範例測資，除數、被除數交換

```
┌─yuchen0515@NTNUMATHLIN ~/程式設計hw6
└─$ ./hw0605
Quotient: 0
Remainder: 12171
```

測資 2 除數大於被除數，商應為 0，餘數就是被除數

說明

6 Bonus: Printf (10 pts)

Please run the following code and give an explanation.

```
1  #include <stdio.h>
2  #include <unistd.h>
3
4  int main()
5  {
6      printf( "Hello World" );
7
8      sleep( 10 );
9
10     return 0;
11 }
```

◎**題意說明**→ 請實際運行以上的範例，並解釋他

Tip: 其實在作業五第四題的跑馬燈已遇過相同的問題，在 printf、scanf、fprintf 中我們會用到三種標準串流(*FILE)[根據 Man 所寫]，stdin[standard input]、stdout[standard output]、stderr[standard error]，當時寫跑馬燈時，會有數字不立即印出的狀況，如果使用 fprintf(stderr, "……");，就能「**立即**」印出，假設使用 printf，相當於使用 fprintf(stdout,"…….");，當遇到換行時才會印出(第二種解決方式)。

◎解釋

Printf 是預設使用"stdout"，也就是 standard output(標準輸出)，他是一個定義在<stdio.h>的 macro，今天若將第 6 行改為 fprintf(stderr, "Hello World");，使用的是 standard error(標準錯誤輸出串流)，就能夠先顯示 Hello World 再停止十秒了，不過為何會這樣呢？

源自於 stderr(**unbuffered**), stdout 緩衝區模式的處理是完全不同的，stdout 通常使用的是 **line buffered**，也就是說他有一個緩衝區負責儲存資訊，當他遇到"\n"時才會將 buffer 內的東西 dump 出來，如果沒使用\n 而要印出東西，則函式內必須 flush 你的 buffer 才能把東西丟出來，或著主函式結束也會自動做這件事情。

今天 time 的 sleep 內並不會自動 flush buffer 內的東西，所以到了第 6 行，其實只是緩衝區內存著"Hello World"這幾個字樣，遇到第 8 行，因為 sleep 並不會 flush，因此直接執行他停了十秒，**當程式結束時，會自動 flush buffer**，因此在結束時，"Hello World"就印出來了，因此在第六行只是在 buffer 內存了這些字樣，繼續往後執行，直到程式主動去 flush 才把存在 buffer 內的東西印出來。

如果在第七行加入 scanf，因為 scanf 會先 flush buffer 的東西，所以此時 Hello World 就會先出現，才停止 10 秒了，如下圖的測試：

```c
#include <stdio.h>
#include <stdint.h>
#include <unistd.h>

int main(){
    int32_t t=0;

    printf("Hello World");
    scanf("%d", &t);

    //printf("\n");
    //puts("");
    //fprintf(stderr,"Hello World");
    //stdI/O

    //fprintf(stderr, "you are");
    sleep(5);
    //fprintf(stdout, " superman ");
    //fprintf(stderr, "or not.");

    return 0;

```

```
┌─yuchen0515@NTNUMATHLIN ~/程式設計hw6
└─$ ./hw0606
Hello World
```

```
┌─yuchen0515@NTNUMATHLIN ~/程式設計hw6
└─$ ./hw0606
Hello World32
```

(先印出 Hello World 才讓使用者輸入，輸入完停止五秒才結束程式)

另 stderr 因為是"沒有緩存區"的方式，會直接把東西印到螢幕上，因此不會有這樣的問題，基本上它是直接對應到螢幕，直接從標準輸入檔案中得到輸入資料，將「正常」輸出資料輸出到標準輸出檔案，而將錯誤的資訊送到標準錯誤檔案(所以才叫 stderr, error 之意)。

接著來個強烈一點的例子：

```c
#include <stdio.h>
#include <stdint.h>
#include <unistd.h>

int main(){

    printf("Hello World");

    //printf("\n");
    //puts("");
    //fprintf(stderr,"Hello World");
    //stdI/O

    fprintf(stderr, "you are");
    sleep(5);
    //fprintf(stdout, " superman ");
    //fprintf(stderr, "or not.");

    return 0;

}
~
"hw0606.c" 21L, 295C written
```

左圖先"Hello World"，第 14 行 fprintf(stderr, "you are");，再停止五秒。

在傳統上我們會認為，最終印出結果一定是 Hello Worldyou are，即便是先停止五秒還是後停止五秒，輸出都應該是如此？

不過實際輸出結果如下：

```
┌─yuchen0515@NTNUMATHLIN ~/程式設計hw6
└─$ ./hw0606
you are
```

"先印出 you are"→停止五秒→"Hello World"

```
└─$ ./hw0606
you areHello World
```

為何反而是 you are 先在前面呢？

從上述原理講起，在第七行，我們不印出，而是先將 Hello World 丟到 buffer 內儲存，到了第 14 行，因 stderr 是”不使用緩存區”直接印出，所以先印出了 you are，到了第 15 行停止五秒，最終結束程式後，因為會自動 flush，因此將 buffer 內的東西吐出來印出 Hello World。

下列寫法會更加明顯：

```c
#include <stdio.h>
#include <stdint.h>
#include <unistd.h>

int main(){

    printf("Hello World");

    //printf("\n");
    //puts("");
    //fprintf(stderr,"Hello World");
    //stdI/O

    fprintf(stderr, "you are");
    sleep(5);
    fprintf(stdout, " superman ");
    fprintf(stderr, "or not.");

    return 0;

}
~
"hw0606.c" 21L, 289C written
```

第 16 行的 fprintf(stdout)相當於 printf。

按照上述邏輯，在 buffer 內應是第七行時存入 Hello World，到第 14 行先印出 you are，之後停止五秒後，buffer 內再多存 superman，第十七行直接印出 or not.

因此第一步，螢幕上印出的是 you areor not.

程式結束時，會將 buffer 內的”Hello Worldsuperman”吐出來，因此最終結果應該是”you areor not.Hello Worldsuperman”

```
┌─yuchen0515@NTNUMATHLIN ~/程式設計
└─$ ./hw0606
you areor not.Hello World superman
```

那假設我在第十六行加入\n 呢?

執行結果如圖：

```
you areHello World superman
or not.%
```

很弔詭的不是 you are superman or not.Hello World，反而是這樣的結果

因為第十四行，我先印出 you are，此時 buffer 因為第七行的緣故，已經存了 Hello World，停止五秒後，在第十六行 buffer 直接在後面追加 superman，因此現在 buffer 有”Hello World superman”，但因為遇到\n 了，所以把 buffer 的東西全部印出來，到了第 17 行直接印出 or not，因此結果如圖。

備註：使用 wsl(window for linux system 配合 cmder)時，有時會在句末出現”%”