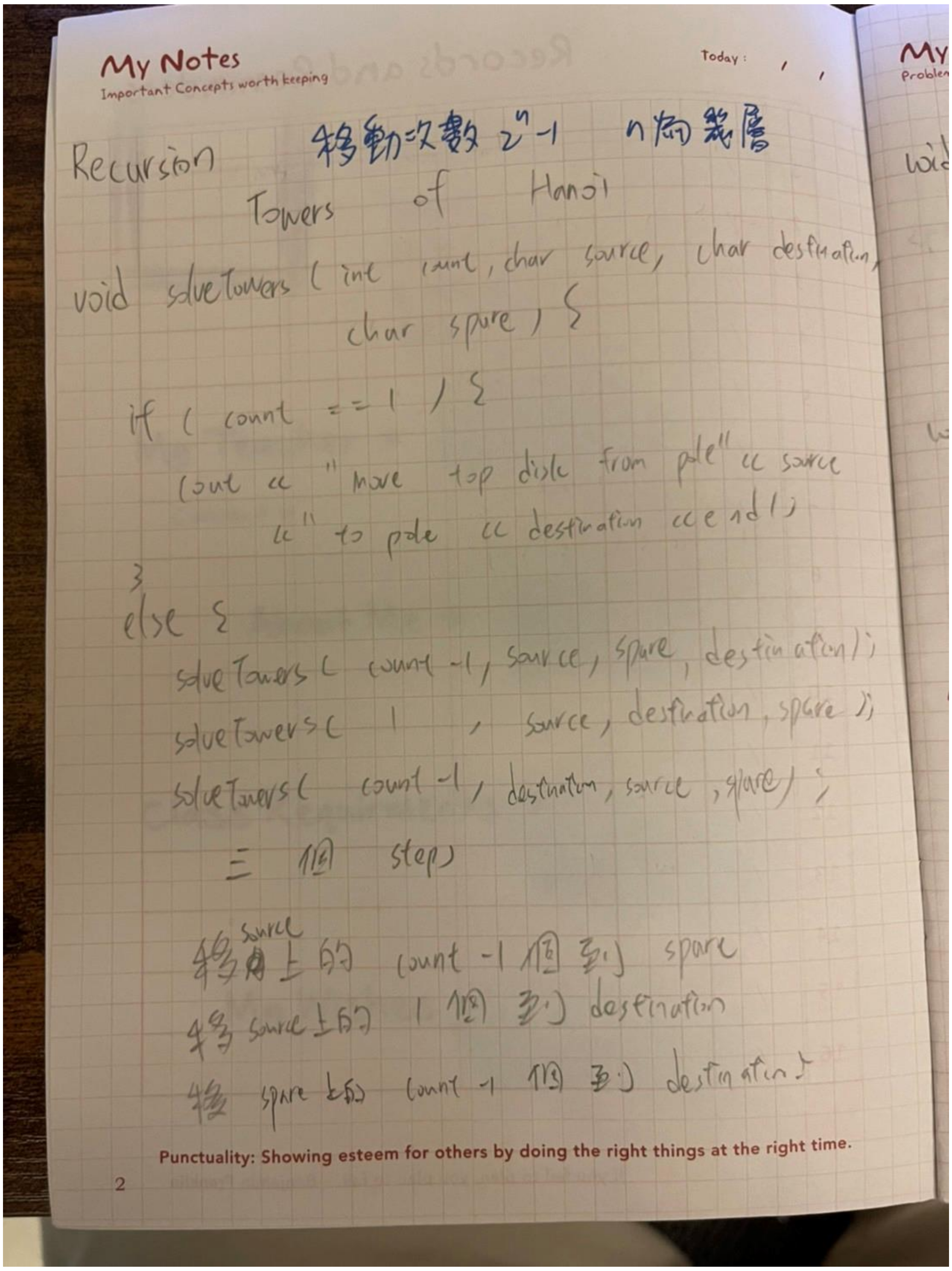


My Notes
Important Concepts worth keeping

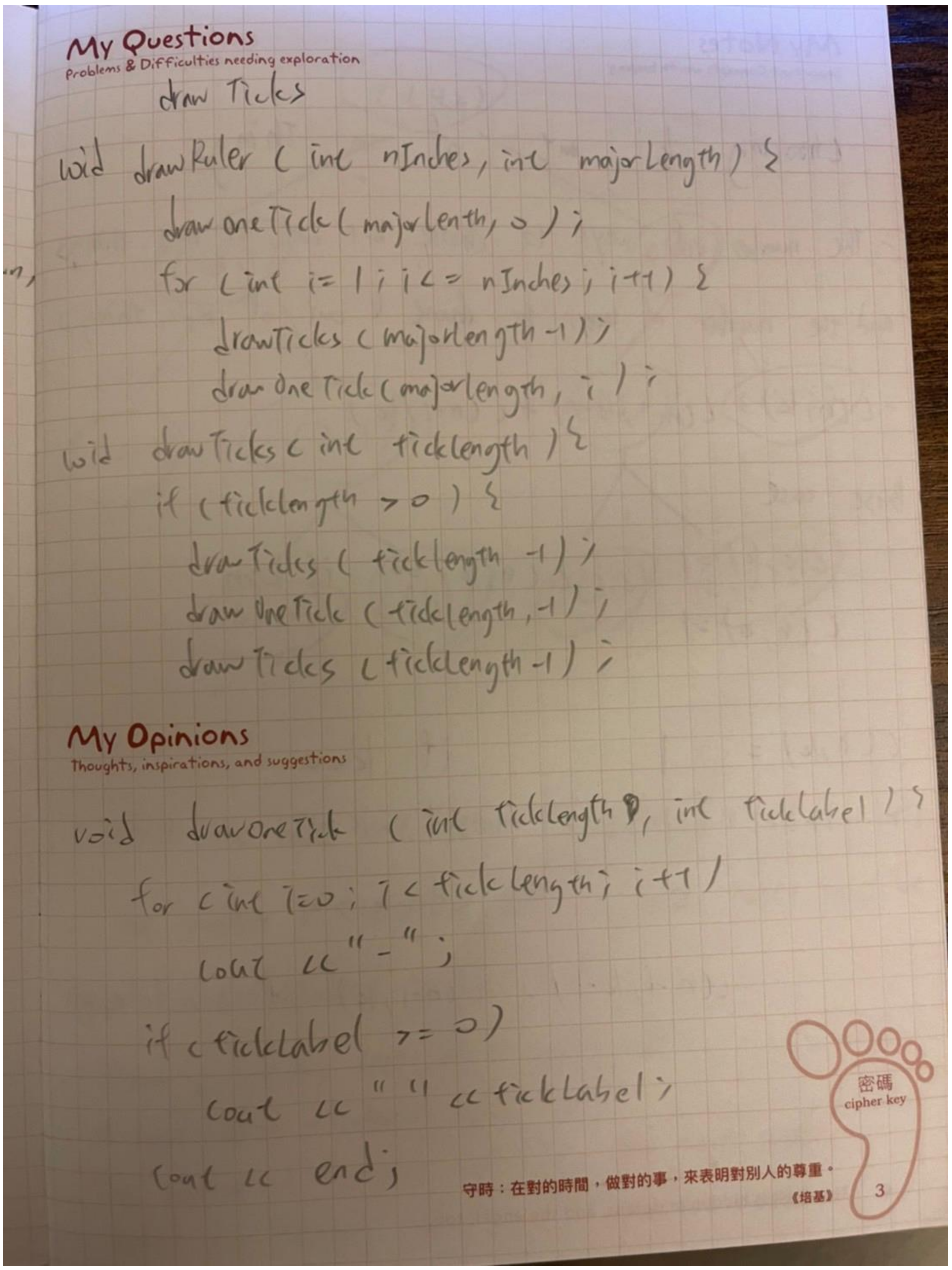
Today : , ,

Recursion 移動次數 2^n-1 n為幾層

Towers of Hanoi

```
void solveTowers( int count, char source, char destination,
                  char spare ) {
  if ( count == 1 ) {
    cout << "move top disk from pole" << source
         << " to pole << destination << endl;
  }
  else {
    solveTowers( count-1, source, spare, destination);
    solveTowers( 1      , source, destination, spare );
    solveTowers( count-1, destination, source, spare) ;
      三 個 step)
```

移source上的 count-1 個 到 spare
移 source上的 1 個 到 destination
移 spare上的 count-1 個 到 destination }



My Questions

Problems & Difficulties needing exploration

```
    draw Ticks

void drawRuler ( int nInches, int majorLength ) {
    draw oneTick ( majorlenth, 0 );
    for ( int i = 1 ; i <= nInches ; i++ ) {
        drawTicks ( majorLength -1 );
        draw OneTick ( majorlength , i ) ;
void drawTicks ( int ticklength ) {
    if ( ticklength > 0 ) {
        drawTicks ( ticklength -1 ) ;
        draw OneTick ( ticklength , -1 ) ;
        draw Ticks ( tickLength -1 ) ;
```

My Opinions

Thoughts, inspirations, and suggestions

```
void drawOneTick ( int tickLength, int tickLabel ) {
    for ( int i=0 ; i < ticklength ; i++ )
        cout << "-" ;
    if ( tickLabel >= 0 )
        cout << " " << tickLabel ;
    cout << endl ;
```

密碼
cipher key

Choosing k out of n things

- The number of ways to choose k-1 out of n-1 things and the number of ways to choose k out of n-1 things

- $C(n,k) = C(n-1,k-1) + C(n-1,k)$

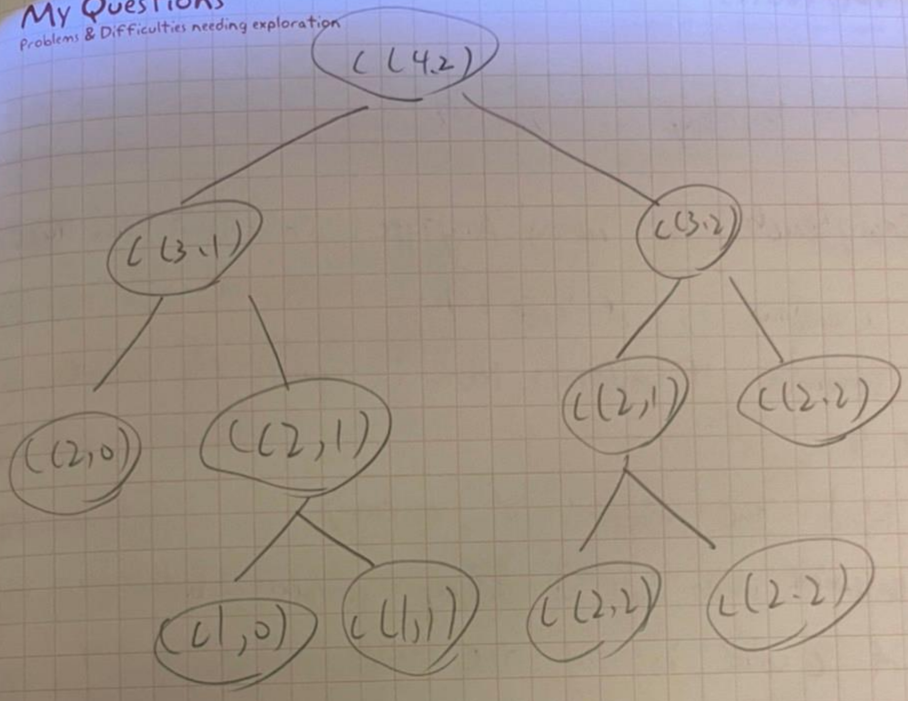
Base case

$C(k,k) = 1$ $C(n,k) = 0$ if $k>n$

$C(n,0) = 1$

$C(n,k) =$
1 if $k=0$
1 if $k=n$
0 if $k>n$
$C(n-1,k-1) + C(n-1,k)$ if $0<k<n$

My Questions
Problems & Difficulties needing exploration



My Opinions
Thoughts, inspirations, and suggestions

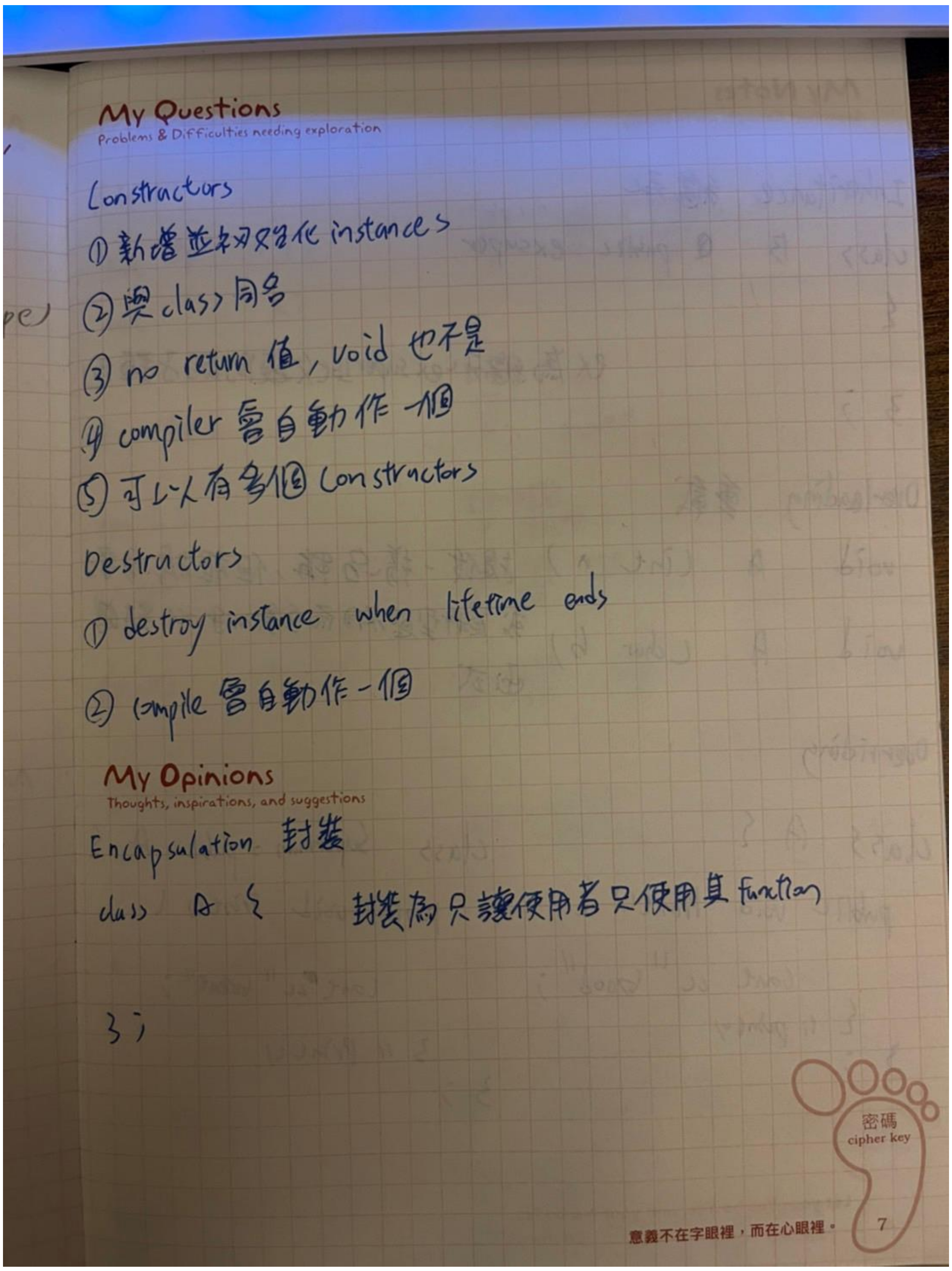
- Since the base case return 1's $c(n,k) = 1+1+1 \cdots +1$
- The number of recursive calls to base cases must be equal to |leaf nodes| = $c(n,k)$
- Draw a binary tree of recursive calls
  * |leaf nodes| - |internal node| = 1
- The number of recursive calls to non-base cases is equal to |internal nodes| = $c(n,k)-1$

密碼
cipher key

魔鬼躲在細縫裡，天使更是這樣。

Binary Search

```
binarySearch ( in anArray : ArrayType , in value : Item Type)
if ( anArray is of size 1 )
    Determine if anArray's item is equal to value
else {
    Find the midpoint of anArray
    Determine which half of anArray contains value
    if ( value is in the first half of anArray )
        binarySearch ( first half of anArray , value )
    else
        binarySearch ( second half of anArray , value )
}
```



My Questions

Problems & Difficulties needing exploration

Constructors

① 新增並初始化 instances

② 與 class 同名

③ no return 值, void 也不是

④ compiler 會自動作一個

⑤ 可以有多個 Constructors

Destructors

① destroy instance when lifetime ends

② compile 會自動作一個

My Opinions

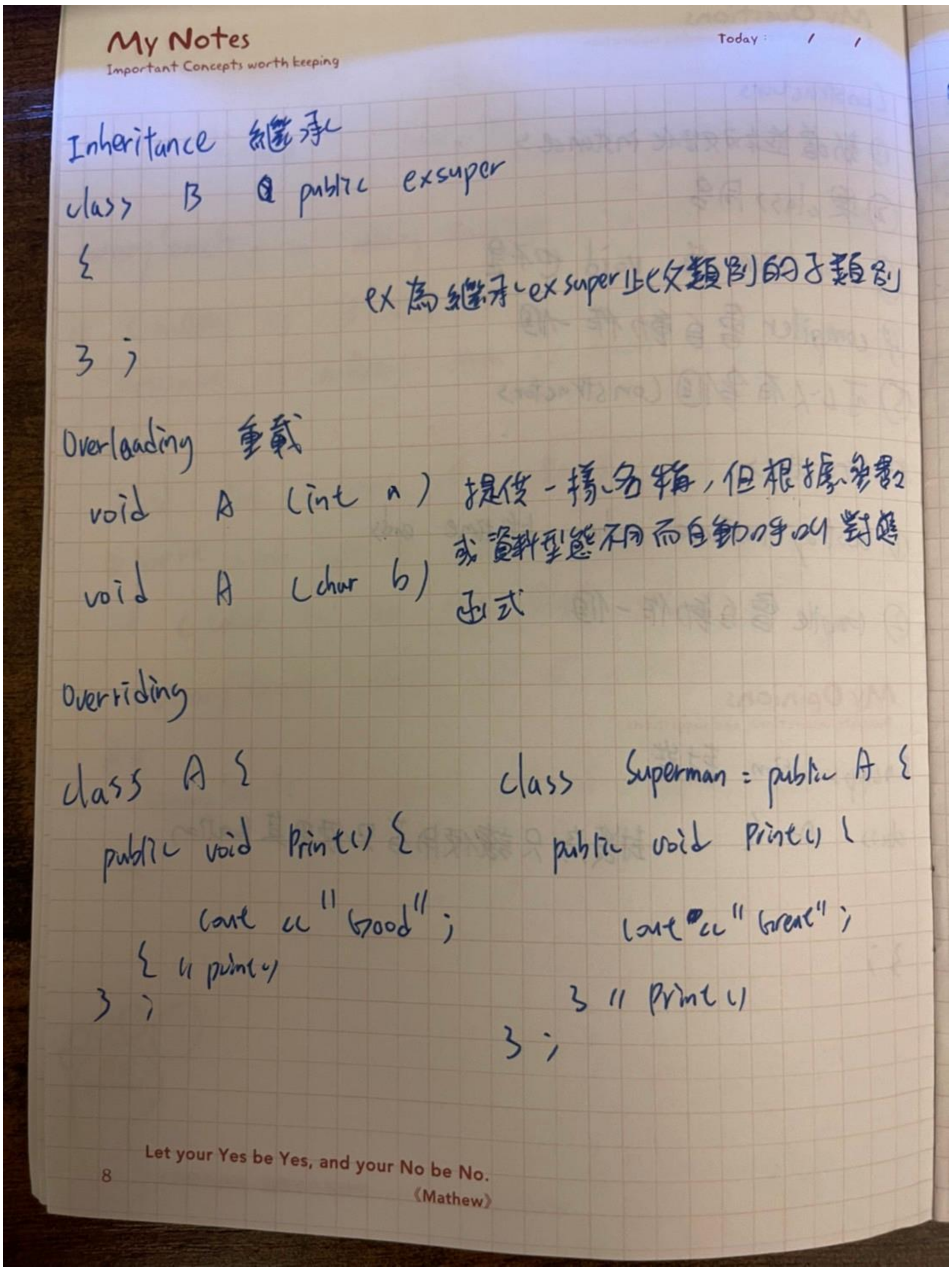
Thoughts, inspirations, and suggestions

Encapsulation 封裝

```
class A {        封裝為只讓使用者只使用其 function

};
```

密碼
cipher key



Inheritance 繼承

```
class B : public exsuper
{

};
```

ex 為繼承 exsuper 此父類別的子類別

Overloading 重載

```
void A (int a)
void A (char b)
```

提供一樣名稱，但根據參數或資料型態不同而自動呼叫對應函式

Overriding

```
class A {
  public void Print() {
        cout << "Good";
  } // print()
};
```

```
class Superman : public A {
  public void Print() {
      cout << "Great";
    } // Print()
};
```

My Questions
Problems & Difficulties needing exploration

Pointers

```
int *p  *q;
int      x;

p = &x;

*p = 6;
p = new int;

*p = 7;
```

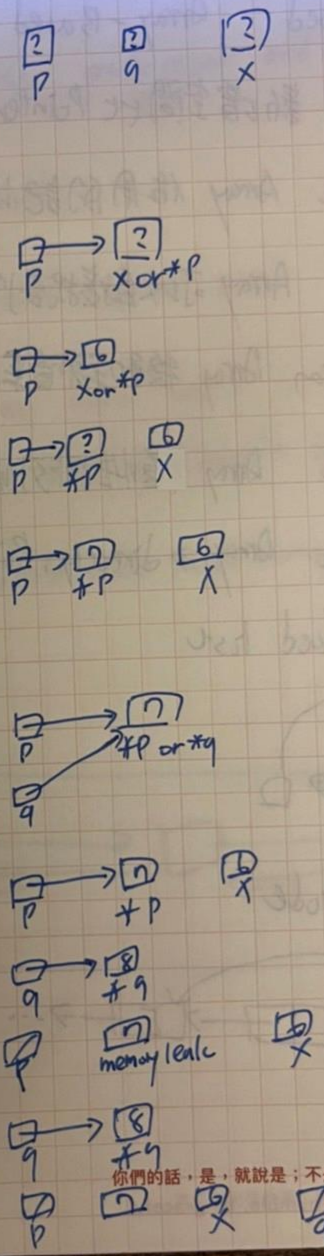
My Opinions
Thoughts, inspirations, and suggestions

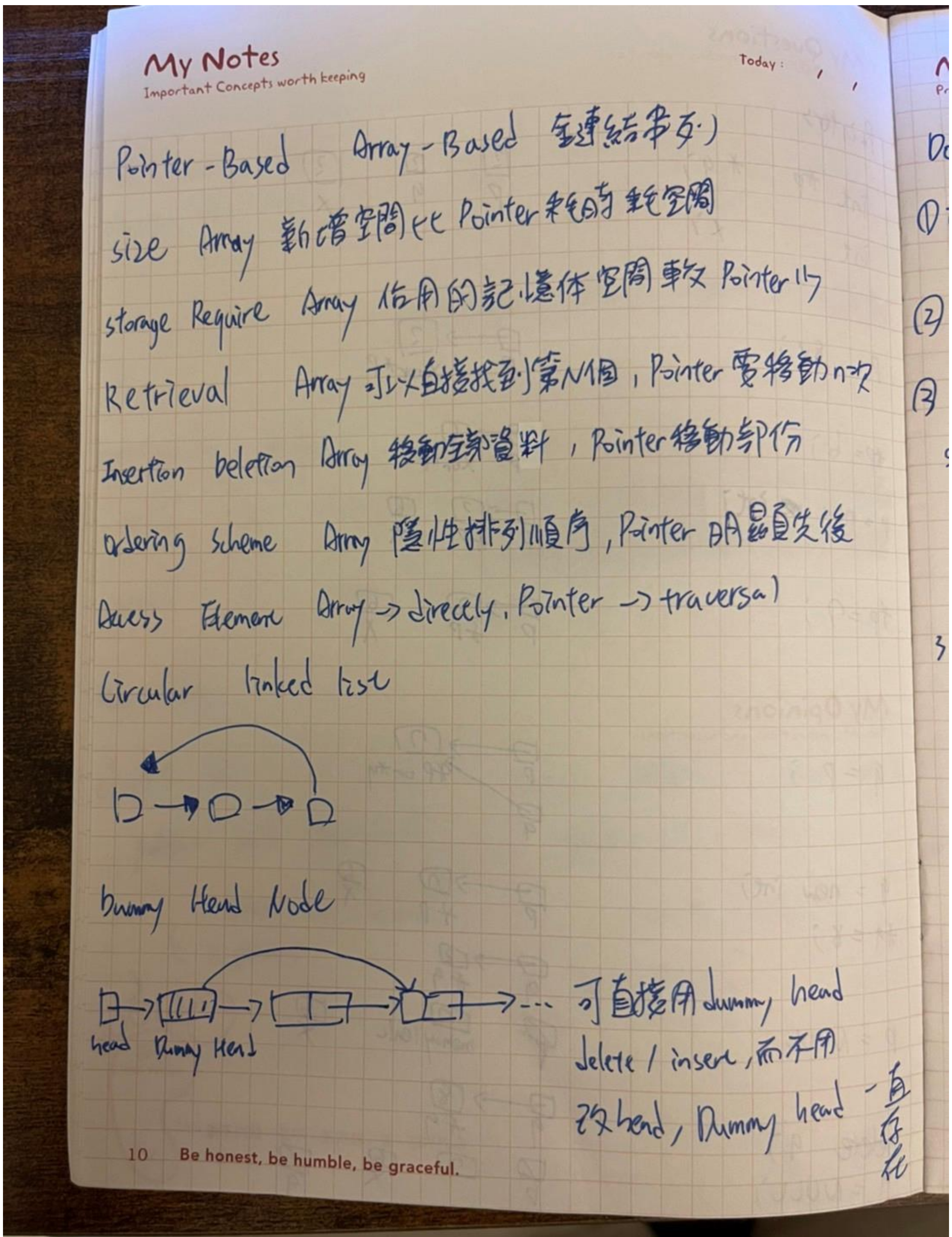
```
q = p;

q = new int;
*q = 8;

p = NULL

delete q;
q = NULL;
```





My Notes

Important Concepts worth keeping

Today: , ,

Pointer-Based Array-Based 鏈結串列)

size Array 新增空間比 Pointer 耗時耗空間

storage Require Array 佔用的記憶体空間較 Pointer 少

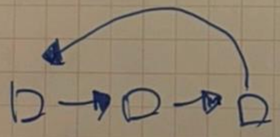
Retrieval Array 可以直接找到第N個, Pointer 要移動n次

Insertion Deletion Array 移動全部資料, Pointer 移動部份

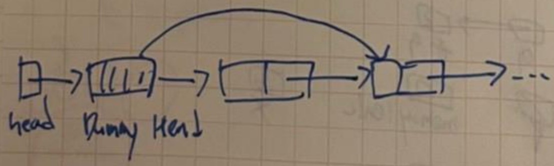
ordering scheme Array 隱性排列順序, Pointer 明顯先後

Access Element Array -> direcely, Pointer -> traversal

Circular linked list



Dummy Head Node



可直接用 dummy head delete / insert, 而不用改 head, Dummy head 一直存在

My Questions
Problems & Difficulties needing exploration

Doubly Linked lists

① Each node points to its predecessor and its successor
前者 (後者)

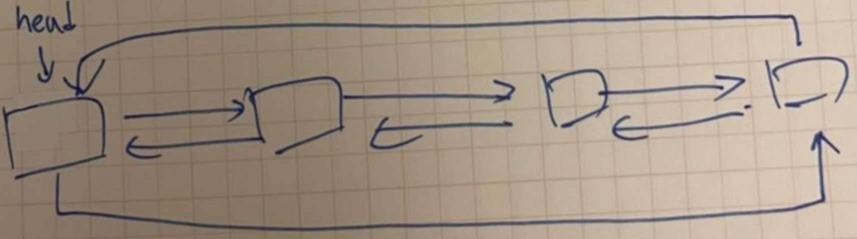
② often has a dummy head

③ often circular to eliminate special cases

struct Node {
  int item
  Node* precede
  Node* next
}; // end Node

有 dummy head node:
dummy head 的 precede 指向尾端
尾端的 next 指向 dummy head

My Opinions
Thoughts, inspirations, and suggestions



密碼
cipher key