My Notes
Important Concepts worth keeping

Today: , ,

Greatest Common Divisor (最大公因數)

- A recursive definition of GCD

① $gcd(x,y) = x$ if $(y=0)$

$= gcd(x, y \bmod x)$ if $(y>x)$

$= gcd(y, x \bmod y)$ if $(x>y)$

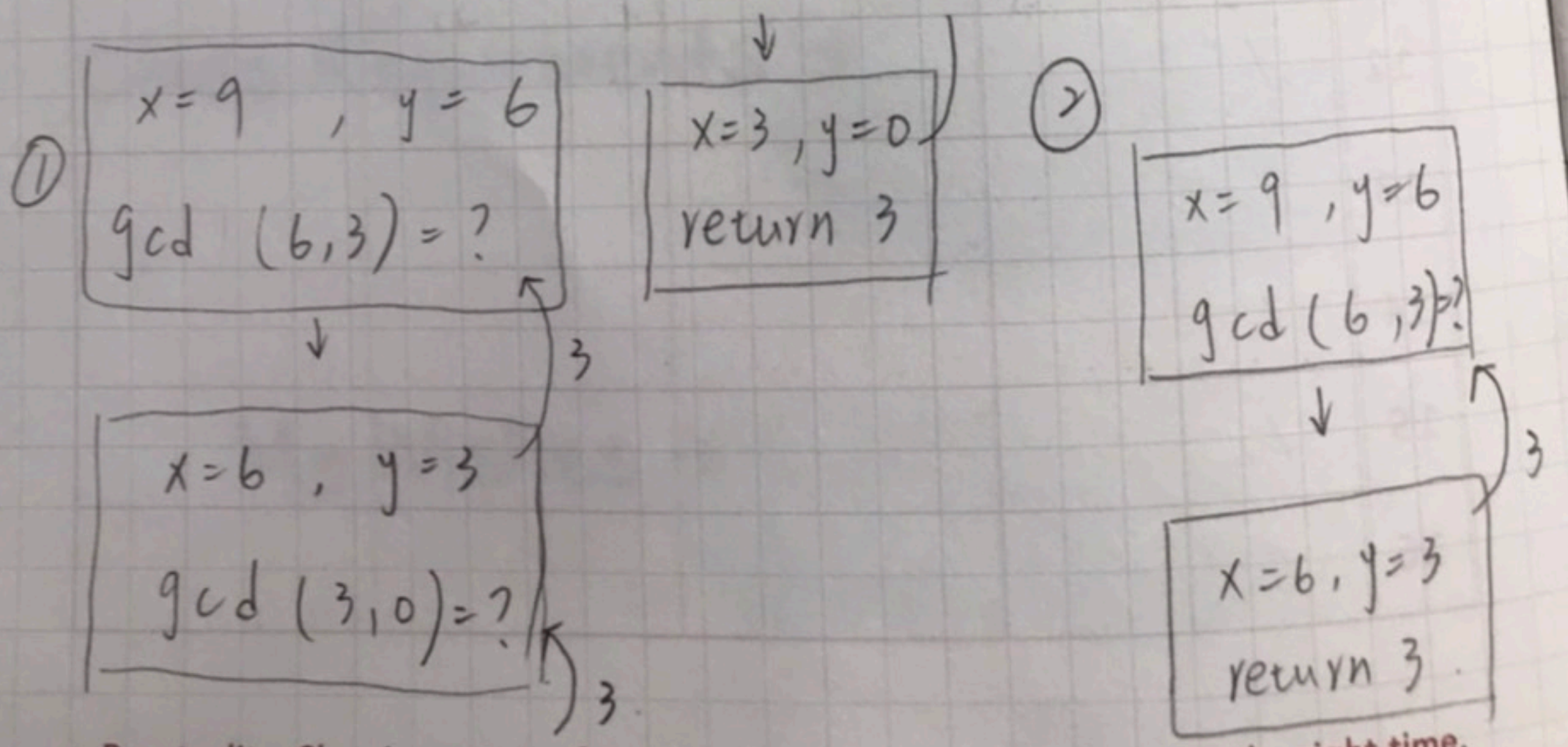
(使越小 數值越來)

② $gcd(x,y) = y$ if $x \bmod y = 0$ ex 120 & 20.

$= gcd(y, x \bmod y)$ otherwise

↘ (輾轉相除法).

- ☆在 $x>=y$ 時效率比較好為 ②



My Quest
Problems & Difficul

Towers

個數

△ 小

例

⇒

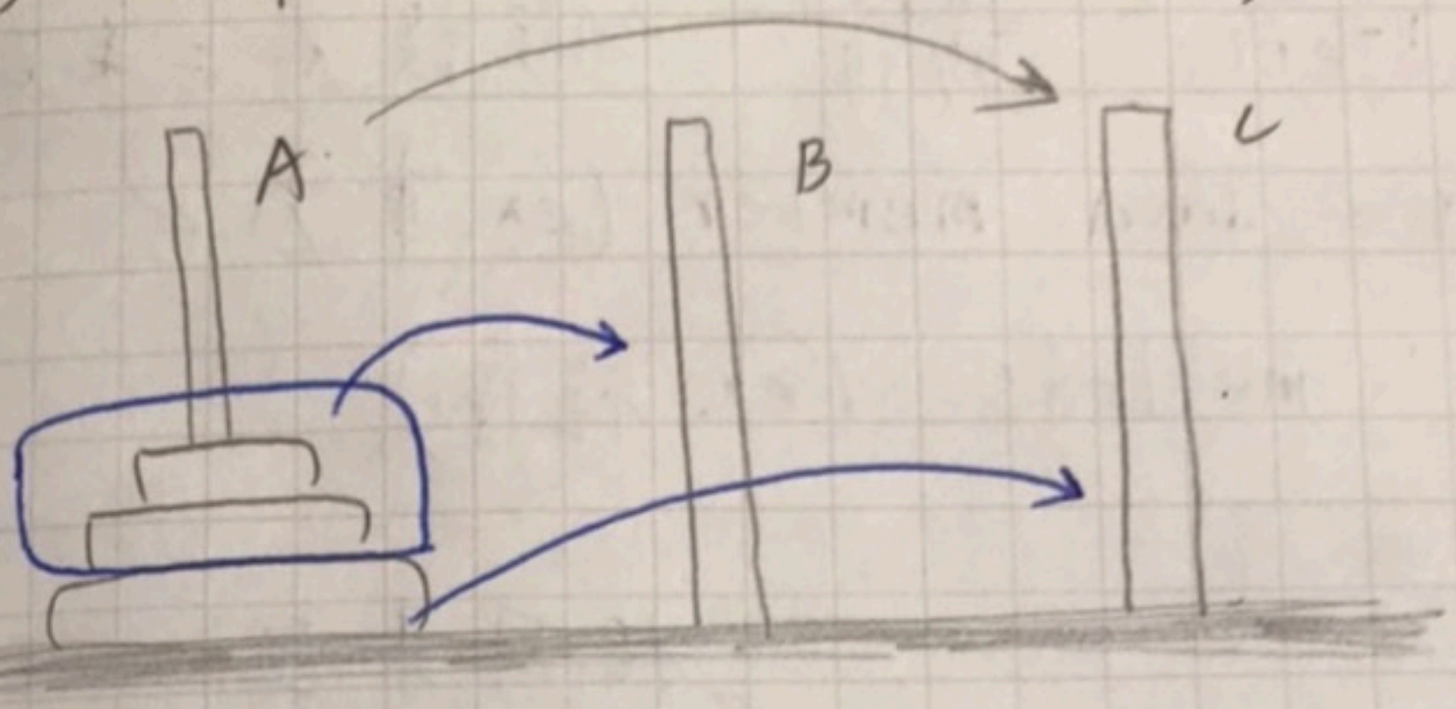
∴

Punctuality: Showing esteem for others by doing the right things at the right time.

My Questions
Problems & Difficulties needing exploration

My Opinions
Thoughts, inspirations, and suggestions

Towers of Hanoi (河內塔)



個數

Source 起點　Auxiliary 輔助　Destination 終點

△小的必須在最上面，不可被比它大壓住

例3個盤子

⇒ 先把2個搬到B，剩下搬到C

∴ 假設為100個，64，32 都為 個數-1 搬入B.

公式 ⇒ $2^{\text{幾個盤子}} - 1 =$ 次數

My Notes
Important Concepts worth keeping

物件導向

Today : , ,

- class of object ← 同類的物件
  - Attributes : data member (ex. 車)
  - Behaviors : methods (ex. 程序)

  ◦假設為二手車商

↓

* three characteristics
  - Encapsulation 封裝 (hides inner details) → 相同東西 可更有效率，不受外界影響
  - Inheritance 繼承 (reused = copy) 針對"不同"東西重寫
  - Polymorphism 多型

Abstract Data Types (ADT) 抽象化

- Modularity 模組化 (容易寫、讀、修改 modify)

Achieve a Better Solution

* cohesion 高內聚

儘可能讓模組只做一件事 (不只一件事)

* coupling 低耦合

之間的關係不大，所以把它們獨立

My Questions
Problems & Difficulties needing exploration

My Opinions
Thoughts, inspirations, and suggestions

- Functional abstraction 功能性的抽象化
  - → 不須考慮效率(如何敘述)
  - * specifications 描述
  - * implementation 實作) 分開的兩件
    - → .cpp
    - ↑ Information hiding 資料隱藏.

- 可把不同模組分開 (Abstract Data Types).

Implementing ADTs (實作)

- 以 C++ 為例 (c++ classes).
  - → An object's data and methods are encapsulated. ↗封裝
  - → 可設為 Public or (Private) → 外界動不到.
  - * constructors (建構-空間, 佔領)
  - → 每個 class 都必須要有!!!
  - * Destructor (預設解構)
  - → 相對於 constructors.

Inheritance in C++

ex. class Colored sphere : public Sphere
小 大
↑ derived ↑ base class
子類別 class 父類別

My Notes
Important Concepts worth keeping

鏈結串列

Today: , ,

△基本架構（有一個頭，最後有門擋住）

△☆Programming with Linked List 指標.（主要）

△☆pointer - based Implementations 鏈結串列

- Pointers ⇒ contains the location, or address

  指標 = 門牌 (int *p)

- *p ⇒ memory cell

- p = &x ⇒ &x = 房子x的門牌

- 必須先新增 p = new int;

  ↓

  發現記憶體配置錯誤 bad_alloc

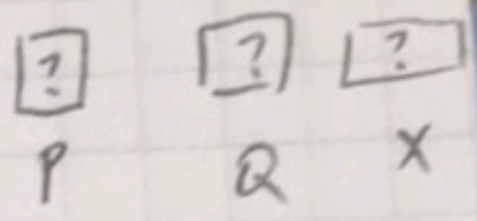
- run完一次要 delete，再設為 NULL

  ↓

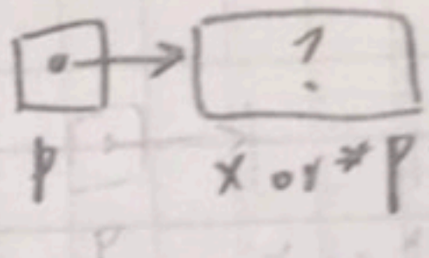
  因可能會誤用

例子：(以指標為基礎)

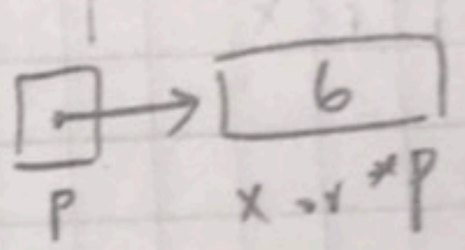
(a) Declaring pointer `int *P, *Q;` `int X;`
Variable



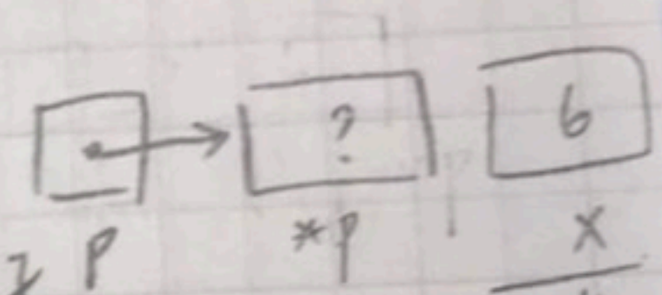
(b) pointing to statically allocated memory `P = &X;`



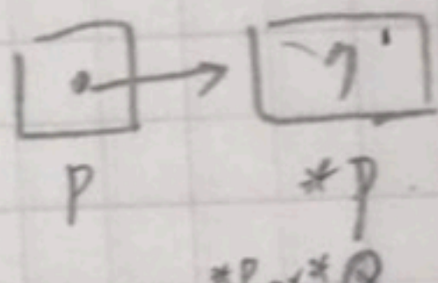
(c) assigning a value `*P = 6;`
↳房子內容



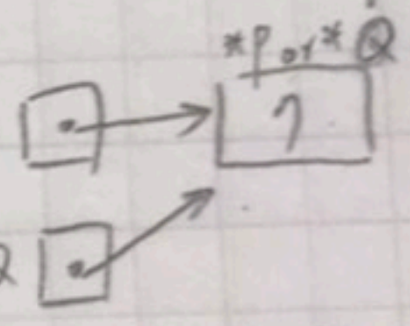
(d) allocating memory dynamically `P = new int;`
↳擁有existed的房子



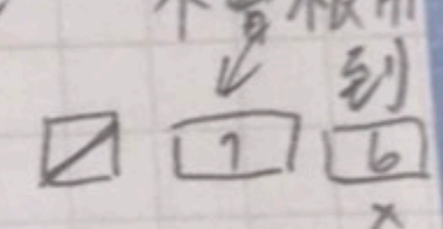
(e) assigning a value `*P = 7`



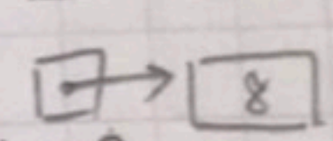
(f) copying a pointer `Q = P`



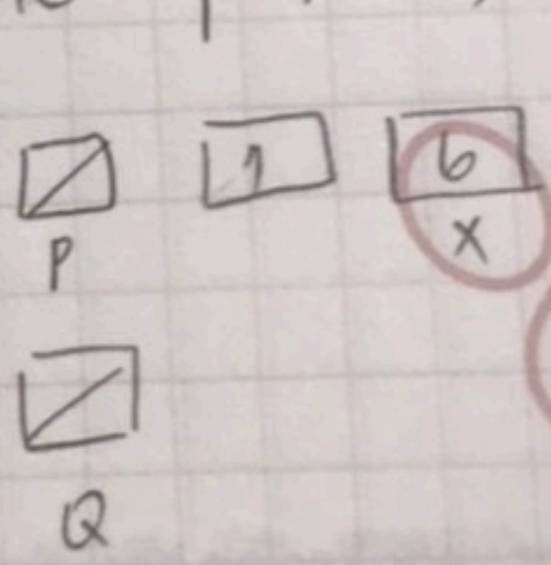
(g) allocating memory dynamically and assigning a value `Q = new int` `*Q = 8`



(h) assigning NULL to a pointer variable `P = NULL;`



(i) deallocating memory `delete q;` `q = NULL;`



(a)~(h) 不可對調

錯誤寫法，(i)才對

memory leak

要先delete.