

# สวัสดีครับ

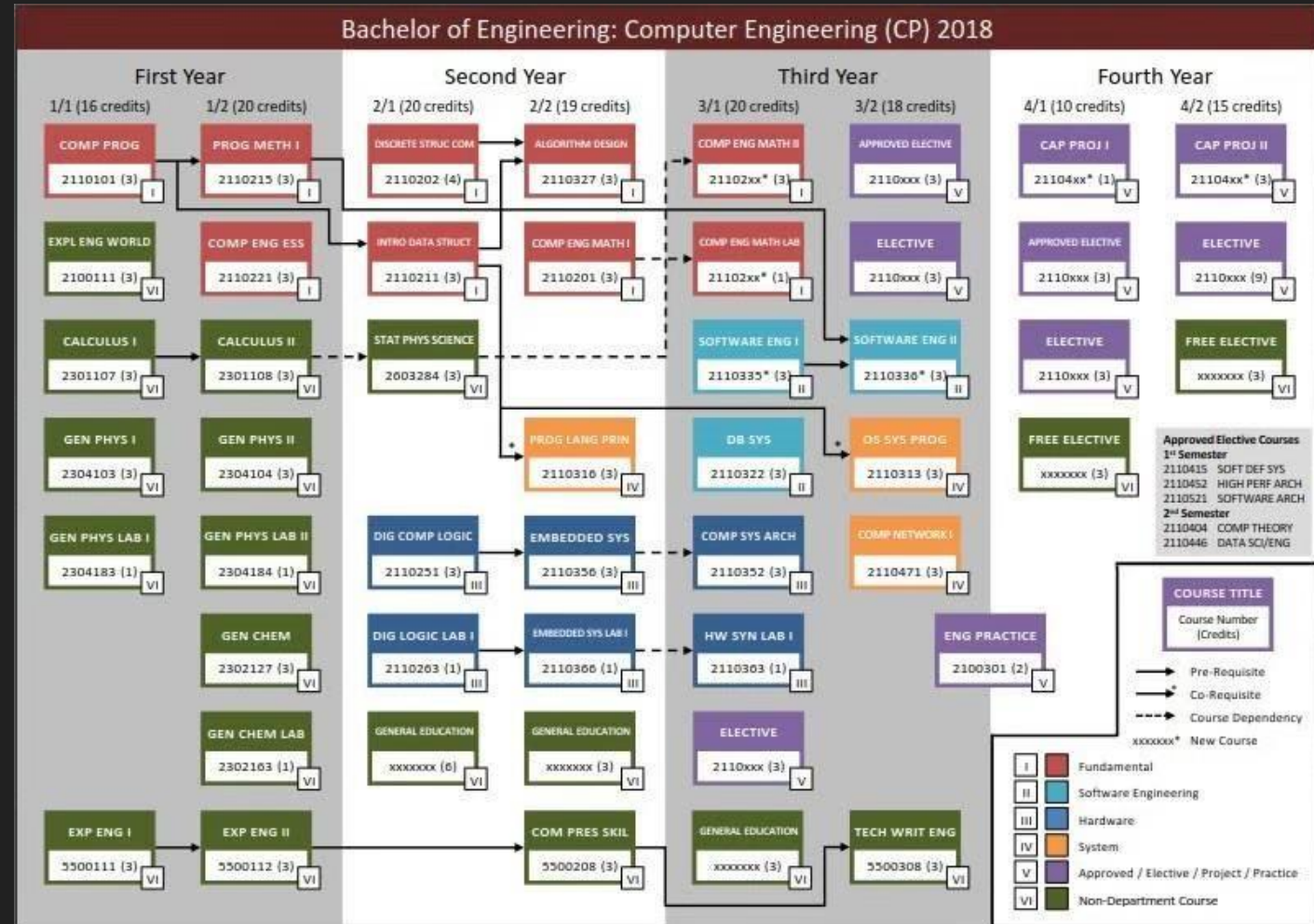
- ระหว่างรอคนอื่น ๆ ใครยังไม่ได้เข้า Discord ของวิชา (CP48 Learning) รบกวนเข้าได้เลยครับ
  - <https://discord.gg/vaFEarNVDD>
- อย่าลืม join myCourseVille วิชานี้
  - Password: binary

# Introduction to Data Structure

2565 เทอมต้น

# วิชานี้ ในหลักสูตร

- อยู่ในกลุ่ม Fundamental
- เน้นเขียนโปรแกรม
- เป็น prerequisite ของ 2 วิชา
  - Algo Design
  - Prog Lang Prin



# เรียนไปทำไม?

## ใช้งาน Data Structure

- เลือกใช้ Data Structure ในการแก้ปัญหาต่าง ๆ ได้
  - เข้าใจข้อดี ข้อด้อย
  - รู้ว่าปัญหาแบบไหนควรใช้อะไร
- วิเคราะห์ ประสิทธิภาพเชิงเวลา ประสิทธิภาพเชิงพื้นที่ ได้
  - รู้ว่าทำอะไรแล้วเร็ว หรือ ช้า

## สร้าง Data Structure

- เข้าใจหลักการของ Data Structure
- เข้าใจหลักการทำงานพื้นฐานของการเก็บข้อมูลในคอมพิวเตอร์
- ประยุกต์ ดัดแปลง Data Structure ให้เหมาะกับการใช้งาน

ภาษา C++

# เรียนกันอย่างไร?

- มี video เนื้อหา และ บันทึกกิจกรรม ให้ดู
- เรียน on-site แน่ ๆ ในเดือน สิงหาคม เดือนอื่น ๆ ดูตามสถานการณ์
- สอบ on-site แน่ ๆ
- ใน lecture แต่ละครั้ง จะแบ่งเป็นสองกลุ่ม
  - กลุ่มแรก สำหรับคนที่ยังดู video ไม่ครบ จะมานั่งดูกัน เป็น lecture
  - กลุ่มสอง สำหรับคนที่ดูแล้ว จะเป็นการลงทำโจทย์หรือถามตอบในเนื้อหา
    - “มักจะ” มีการเขียนโปรแกรมในห้องเรียน นำ notebook มาด้วยจะสะดวกกว่ามาก ๆ

# ความคาดหวังของผู้สอน

- ดู video “ก่อน” เข้าเรียน
- ถ้าสงสัย อย่าเก็บไว้ ถามเลย
  - ในห้องเรียน
  - ใน Discord
  - email ก็ได้
- ทำโจทย์ใน grader
- ซ้อมทำโจทย์ midterm/final

วิชานี้ 3-0-6 จริง ๆ  
คือ “ต้อง” ใช้เวลา  
นอกห้องเรียนด้วย

# เนื้อหาที่เรียน

- เดือนแรก: ใช้งาน
- เดือน 2-4: สร้าง
  - Memory Management
  - Pointer
  - Linear Structure
  - Tree Structure

อย่าลืมดู Syllabus ใน mCV

วันที่	หัวข้อ	วัดใจ
8 ส.ค.	Intro to Data Structure	
10 ส.ค.	Introduction to C++: IDE, compiler, simple	
15 ส.ค.	<u>std::vector</u> , "Word Count Problem", <u>std::set</u> , <u>std::pair</u> , temp	
17 ส.ค.	More on <u>std::vector</u> , vector iterator,	
22 ส.ค.	basic usage of <u>std::map</u> , iterator, <u>std::sort()</u> , <u>std::find()</u> vs <u>map.find()</u>	
24 ส.ค.	Sample QUIZ: Usage of STL	
29 ส.ค.	<u>std::stack</u> , "parenthesis checking"	5-*
31 ส.ค.	<u>std::queue</u> , "radix sort"	6-*
5 ก.ย.	<u>std::priority_queue</u> , "K-th smallest problem", Sorting, custom sorting, operator overloading of "<" and "<="	7-*
7 ก.ย.	Quiz #1: Usage of STL	
12 ก.ย.	Create our own data structure, Implementation of <u>CP::pair</u>	8-*
14 ก.ย.	Implementation of <u>CP::vector</u>	9-*
19 ก.ย.	Implementation of <u>CP::stack</u>	10-*
21 ก.ย.	Implementation of <u>CP::queue</u>	11-*
26 ก.ย.	Complexity Analysis, Measurement of efficiency, Asymptotic Notation (Big-Oh, etc.) 1	12-1
28 ก.ย.	Complexity Analysis, Measurement of efficiency, Asymptotic Notation (Big-Oh, etc.) 2	12-2 to 12-3
6 ต.ค.	สอบกลางภาค	
10 ต.ค.	Quiz #2: Vector and Stack	
12 ต.ค.	Introduction to Binary Heap, Graph & Tree	13-1 to 13-3
17 ต.ค.	Implementation of <u>CP::priority_queue</u>	13-4 to 13-7
19 ต.ค.	Pointer, Linked List	14-1 to 14-3
24 ต.ค.	วันหยุด (จดหมายเหตุวันปิยมหาราช)	
26 ต.ค.	Quiz #3: Queue and Priority Queue	
31 ต.ค.	Implementation of <u>CP::list</u>	14-4 to 14-5
2 พ.ย.	Binary Tree	15-*
7 พ.ย.	Binary Search Tree, Implementation of <u>CP::map_bst</u>	16-*
9 พ.ย.	AVL Tree, Implementation of <u>CP::map_avl</u>	17-*
14 พ.ย.	Introduction to Hash, separate chaining	18-1 to 18-4
16 พ.ย.	Hash, open addressing	18-5 to 18-9
21 พ.ย.	Quiz #4: Pointer, List, Tree	
23 พ.ย.	Problem Session	
7 ธ.ค.	สอบปลายภาค	








# วัตถุประสงค์

- อธิบายหลักการของโครงสร้างข้อมูลในการจัดเก็บข้อมูลแบบต่าง ๆ
- วิเคราะห์ประสิทธิภาพในเชิงเวลาและเนื้อที่ในการทำงานของโครงสร้างข้อมูล
- ประยุกต์ใช้โครงสร้างข้อมูลให้เหมาะสมกับปัญหา
- ออกแบบโครงสร้างข้อมูลที่มีประสิทธิภาพ



# ทำไม?

- พื้นฐานของ Computer Science
- ได้รู้จักภาษาอื่น
  - ใช้ C++ เพราะจะได้เข้าใจการทำงานของคอมพิวเตอร์มากขึ้น
- เขียนโปรแกรมให้คล่อง ๆ

Aug 2022	Aug 2021	Change	Programming Language		Ratings	Change
1	2	▲		Python	15.42%	+3.56%
2	1	▼		C	14.59%	+2.03%
3	3			Java	12.40%	+1.96%
4	4			C++	10.17%	+2.81%
5	5			C#	5.59%	+0.45%
6	6			Visual Basic	4.99%	+0.33%
7	7			JavaScript	2.33%	-0.61%


# เสียงจากรุ่นพี่

นักพัฒนาซอฟต์แวร์รุ่นใหม่ ไม่ค่อย  
เข้าใจ Fundamental

*By Nuuneoi CP28*

*(Software Eng @ Facebook)*

*Founder, Apetivism*

Sittiphol Phanvilai [Follow](#)Mar 21, 2016 · 1 min read[Twitter](#) [LinkedIn](#) [Facebook](#) [Bookmark](#)

เป็นประโยชน์จริงๆที่ได้ยินมา 3 ครั้งรวดภายในวันเดียวจากรุ่นพี่รุ่นเพื่อนรุ่นน้องที่ไปเจอมาที่ตลาดกระบี่และจุกะวันก่อน

มันคงไม่ใช่เรื่องบังเอิญแล้วแหละ ...

ซึ่งก็เห็นด้วยเลย โดยเฉพาะเด็กรุ่นใหม่สายโปรแกรมมิ่ง ทุกวันนี้โยกหาแต่ “วิธีลัด” “ของง่ายๆ” หรือ “วิธีทำให้โค้ดสั้นที่สุด” แต่พอถามว่าโค้ดเหล่านั้นทำงานยังไง ทำไมต้องเขียนแบบนั้น

“ไม่รู้ดีพี่ ก็ทำตามเค้า ก็อปแปะมา...”

ผลคืองานชิ้นนั้นอาจจะใช้งานได้(และเดาว่าจะพังในอนาคต) แต่คนๆนั้นไม่ได้เก่งขึ้นเลย ไม่สามารถต่อยอดผลงานไปไหนได้ ทั้งๆที่หนึ่งในผลผลิตของการพัฒนาซอฟต์แวร์ไม่ได้อยู่ที่ตัวซอฟต์แวร์เท่านั้น แต่เป็น “ตัวคน” ด้วย

# ตัวอย่าง เลือกใช้ data structure

1. (10 คะแนน) ตอบคำถามต่อไปนี้สั้น ๆ ว่า แต่ละปัญหามีที่เก็บข้อมูลประเภทใด

1.0 ต้องการเก็บรายชื่อนิสิตคณะวิศวกรรมศาสตร์ทุก ๆ รุ่น แต่ละรุ่นมีหมายเลขรุ่นกำกับ เพื่อเขียนเมทอด  
`int getClassID(string name)` ที่คืนหมายเลขรุ่นของคนชื่อ `name`

ตอบ:..... `map<string,int>` *key* คือชื่อ *mapped value* คือหมายเลขรุ่น (ข้อนี้เป็นตัวอย่าง).....

1.1. ต้องการเก็บข้อมูล คนติดเชื้อโควิดแต่ละจังหวัด ซึ่งแต่ละจังหวัด จะมีข้อมูลของแต่ละวันที่ (วันเดือนปี) ซึ่งแต่ละวันจะมีจำนวนคนติดเชื้อ จำนวนคนเสียชีวิต และจำนวนคนฉีดวัคซีน ของวันนั้น ๆ โดยจะต้องสามารถหาข้อมูลได้อย่างรวดเร็วด้วยการป้อนชื่อจังหวัด และวันที่ ได้

1.2. ต้องการเก็บข้อมูลหูฟังไร้สาย ข้อมูลจะมี รุ่น, ราคา, review score (อาจมีหูฟังที่คะแนนเท่ากัน), คุณสมบัติ (ซึ่งแต่ละหูฟังมีได้มากกว่าหนึ่งอย่าง เช่น sleep, noise cancel, shape memory ฯลฯ ชนิดของคุณสมบัตินี้มีจำนวนมาก (แต่ไม่เท่าจำนวนรุ่น) และหูฟังหนึ่งๆจะไม่มีคุณสมบัติเกิน 2 อย่าง จะต้องหาหูฟังได้ด้วยการระบุคุณสมบัติ (แต่ระบุได้เพียงอย่างเดียวต่อการหาหนึ่งครั้ง) และโปรแกรมจะลิสต์หูฟังที่มีคุณลักษณะนั้นมาให้ทั้งหมดได้ พยายามให้มีการเก็บข้อมูลซ้ำซ้อนน้อยที่สุด

1.3. จากข้อ 1.2 ถ้าอนุญาตให้มีการเก็บข้อมูลซ้ำซ้อนได้ จะใช้โครงสร้างข้อมูลอย่างไร ถึงน่าจะทำงานได้เร็ว

1.4. ต้องการที่เก็บข้อมูล ร้านขายสัตว์เลี้ยง ที่สามารถค้นหาได้ด้วย ชนิดของสัตว์และชื่อสถานที่ใกล้เคียง (ซึ่งสามารถเจอมากกว่าหนึ่งร้าน) ตัวร้านเอง มีข้อมูล คือ ชื่อร้าน และ คำบรรยายร้าน

1.5. ต้องการเก็บข้อมูลคนรอหน้าร้านอาหารบาร์บีคิวปลาซ่า โดยสิ่งที่ต้องการเก็บคือ หมายเลขที่มามองจองที่นั่งและชื่อของคนจอง เพื่อไว้เรียกและตรวจสอบคนเข้าร้าน

# ตัวอย่าง วิเคราะห์ประสิทธิภาพการทำงาน

(4 คะแนน) ในแต่ละข้อย่อยต่อไปนี้ หากเรากำหนดให้  $n$  มีค่า 1,000,000 จงระบุว่า code ในชุด A หรือ B ที่จะทำงานเสร็จก่อนกัน โดยให้ระบุว่า A หรือ B หรือ เลือกที่จะไม่ตอบก็ได้ หากตอบถูก จะได้คะแนนข้อละ 1 คะแนน หากตอบผิด จะได้คะแนน -0.5 คะแนน แต่ถ้าหากไม่ตอบ จะได้ 0 คะแนน

ข้อ ย่อย	Code A	Code B	คำตอบ
(1)	<pre>vector&lt;int&gt; v(10); while (n--) v.size();</pre>	<pre>vector&lt;int&gt; v(10); while (n--) v.push_back(1);</pre>	
(2)	<pre>map&lt;int,int&gt; m; for (int i = 0;i &lt; n;i++) m[i] = i;</pre>	<pre>map&lt;int,int&gt; m; for (int i = 0;i &lt; n;i++) m[1] = i;</pre>	
(3)	<pre>set&lt;int&gt; s; for (int i = 0;i &lt; n;i++) s.insert(i)</pre>	<pre>priority_queue&lt;int&gt; pq; for (int i = 0;i &lt; n;i++) pq.push(i)</pre>	
(4)	<pre>queue&lt;int&gt; q; for (int i = 0;i &lt; n;i++) {     q.push(i);     q.pop(); }</pre>	<pre>stack&lt;int&gt; s; for (int i = 0;i &lt; n;i++) s.push(i); for (int i = 0;i &lt; n;i++) s.pop();</pre>	

# ตัวอย่าง เขียนโปรแกรมใช้ได้

## Hiatus

มีนักเขียนการ์ตูนรายหนึ่งเขียนการ์ตูนแบบรายเดือน หมายความว่าในแต่ละเดือนนักเขียนจะต้องตีพิมพ์ผลงานการ์ตูน 1 ตอน อย่างไรก็ตามนักเขียนท่านนี้หยุดงานอยู่บ่อย ๆ ทำให้บางเดือนไม่มีการตีพิมพ์ผลงาน ปัจจุบันมีการตีพิมพ์ผลงานมาแล้วทั้งหมด  $n$  ตอน โดยที่เรามีข้อมูลอยู่ว่าในปีไหนเดือนไหน นักเขียนท่านนี้ได้ตีพิมพ์ผลงานบ้าง

เรามีคำถามอยู่  $m$  คำถาม โดยแต่ละคำถามจะระบุข้อมูลปีและเดือนมาให้ เราต้องการทราบว่า ในเดือนนั้นและปีนั้น มีการตีพิมพ์หรือไม่ หากไม่มี ให้ตอบว่าตอนล่าสุดที่นักเขียนดังกล่าวตีพิมพ์ก่อนปีเดือนของคำถามคือตอนของปีเดือนใด  
จงเขียนโปรแกรมเพื่อรับข้อมูล และตอบคำถามดังกล่าว

# ตัวอย่าง ใช้ data structure ในการแก้ไขปัญหาก็ได้

11. (10 คะแนน) จงเขียนฟังก์ชัน `int furthest(vector<int> &v)` เพื่อหาว่ามีคู่ตำแหน่ง `a, b` ไต ๆ ที่ `v[a]` และ `v[b]` มีค่าเท่ากัน และค่า `b-a` มีค่ามากที่สุด (กล่าวคือ เราต้องการหาช่องสองช่องที่มีค่าเหมือนกัน แต่หมายเลขช่องอยู่ห่างกันมากที่สุด) โดยให้คืนค่าเป็นค่าของ `b-a` โจทย์ข้อนี้รับประกันว่าใน `v` มีอย่างน้อย 2 ช่องที่มีค่าเท่ากัน

```
int furthest(vector<int> &v) {
```

## ข้อ 11 (furthest)

เรียก lower\_bound,  
upper\_bound  $\leq 3$

- ถ้ามี bug แบบผิดแน่ ๆ หลากๆ case  $\leq 3$
- $O(n^2 \lg n) \leq 4$
- $O(n^2) \leq 5$ 
  - ถ้ามาพยายาม sort อีกรอบ  $\leq 4$
- ใช้ map เพื่อหาตำแหน่งแรกสุด, ท้ายสุด  $\geq 5$ 
  - ถ้าทำถูกได้ 10
  - ใช้ `map<int,vec>` มีแล้วมาเรียก sort `vec` ได้ 8 (หรือ `map<int,set>>`
    - ไม่ sort แต่วิ่งไล่หา min/max แทนที่จะเอา `front()`, `back()` ได้ 9
    - ใช้ set แต่ใช้ `end()-1` หัก 1 แต้ม
    - ไม่ sort แต่ใช้หัวท้ายเลย ได้ 10
- สร้าง pair `{v[i],i}` แล้ว sort ได้ 10

# ตัวอย่าง ออกแบบโครงสร้างข้อมูลได้

13. (10 คะแนน) เราเที่ยวด้วยกัน!!! รัฐบาลต้องการสนับสนุนการท่องเที่ยว จึงได้จัดทำโครงการคืนเงินค่าตัวเครื่องบินสำหรับการไปพักผ่อนยังโรงแรมต่าง ๆ โครงการนี้เกี่ยวข้องกับคน โรงแรม และเที่ยวบิน โดยมีกฎดังต่อไปนี้
- ให้คนแต่ละคนระบุได้ด้วย “หมายเลขประจำตัวประชาชน” เป็นตัวเลขจำนวนเต็มแบบ int
  - ให้โรงแรมแต่ละโรงแรมระบุได้ด้วย “ชื่อโรงแรม” เป็น string
  - ให้เที่ยวบินถูกระบุได้ด้วย “รหัส” พร้อมกับ “วันที่” โดย รหัสเป็น string และ วันที่ระบุด้วย int
  - เพื่อความสะดวก วันที่ในระบบจะเป็นแบบ int โดยนับเป็นจำนวนวันที่ผ่านมาตั้งแต่ 1 ม.ค. 2563
  - ตัวเครื่องบินและโรงแรมจะเกี่ยวข้องกับ “พื้นที่” ให้สมมติว่าในโครงการนี้มีพื้นที่ที่เป็นไปได้ทั้งหมด  $n$  พื้นที่ พื้นที่แต่ละพื้นที่ระบุได้ด้วยจำนวนเต็มแบบ int
  - รัฐบาลกำหนดไว้ว่าโรงแรมแต่ละแห่งนั้นรองรับพื้นที่ใดบ้าง
  - รัฐบาลจะกำหนดไว้ว่าเที่ยวบินในแต่ละ “รหัส” นั้นรองรับพื้นที่ใดบ้าง (โดยถือว่าเที่ยวบินที่รหัสเดียวกันรองรับพื้นที่เหมือนกันเสมอ ไม่ว่าจะเดินทางในวันใดก็ตาม)
  - นักท่องเที่ยวจะได้เงินค่าตัวเครื่องบินคืนก็ต่อเมื่อพักผ่อนในโรงแรมในวันที่  $x$  และมีการเดินทางด้วยเที่ยวบินในระหว่างวันที่  $x-w$  ถึงวันที่  $x+w$  และ รายการของพื้นที่ที่รองรับโดยเที่ยวบินดังกล่าว และ รายการพื้นที่ที่รองรับโดยโรงแรมดังกล่าวมีอย่างน้อย 1 พื้นที่เหมือนกัน
    - ตัวอย่างเช่น โรงแรม A รองรับพื้นที่ “เชียงใหม่” และ “เชียงราย” ส่วนเที่ยวบิน WD115 รองรับพื้นที่ “เชียงราย” และ “แม่ฮ่องสอน” หากนักท่องเที่ยวพักโรงแรม A ในวันที่ 5 โดยเดินทางด้วยเที่ยวบิน WE115 ในวันที่ 4 (สมมติให้ค่า  $w$  เป็น 1) นักท่องเที่ยวดังกล่าวก็สามารถขอเงินคืนได้
    - แต่ถ้านักท่องเที่ยวเดินทางด้วยเที่ยวบิน FD332 ซึ่งรองรับพื้นที่ “แม่ฮ่องสอน” และ “ลำพูน” ก็จะสามารถขอเงินคืนได้ (เนื่องจากไม่มีพื้นที่อย่างน้อย 1 พื้นที่ที่เหมือนกันระหว่างการพักผ่อนกับการเดินทาง) หรือหากนักท่องเที่ยวเดินทางด้วย WE115 แต่เดินทางในวันที่ 3 หรือ 7 ก็ไม่สามารถขอเงินคืนได้เช่นกัน (เนื่องจากวันที่เดินทางไม่อยู่ในช่วง  $x-w$  ถึง  $x+w$  ของการพักผ่อน)

เราต้องการสร้างระบบสำหรับการจัดการการเงิน โดยสร้างคลาส Travel ซึ่งทำหน้าที่ดังกล่าว โดยคลาสนี้จะต้องรับข้อมูลพื้นที่ที่รองรับของโรงแรมต่าง ๆ และเที่ยวบินต่าง ๆ พร้อมทั้งรับข้อมูลว่า เที่ยวบินแต่ละเที่ยวบินมีใครใช้บริการบ้าง และ โรงแรมแต่ละโรงแรมมีใครเข้าพักในวันใดบ้าง คลาสนี้จะทำหน้าที่คำนวณว่าค่าขอคืนเงินแต่ละค่าขอคืนนั้นสามารถทำได้หรือไม่

คลาส Travel ต้องมีฟังก์ชันต่อไปนี้เป็นอย่างน้อย (นิสิตสามารถเขียนฟังก์ชันอื่นเพิ่มเติมได้)

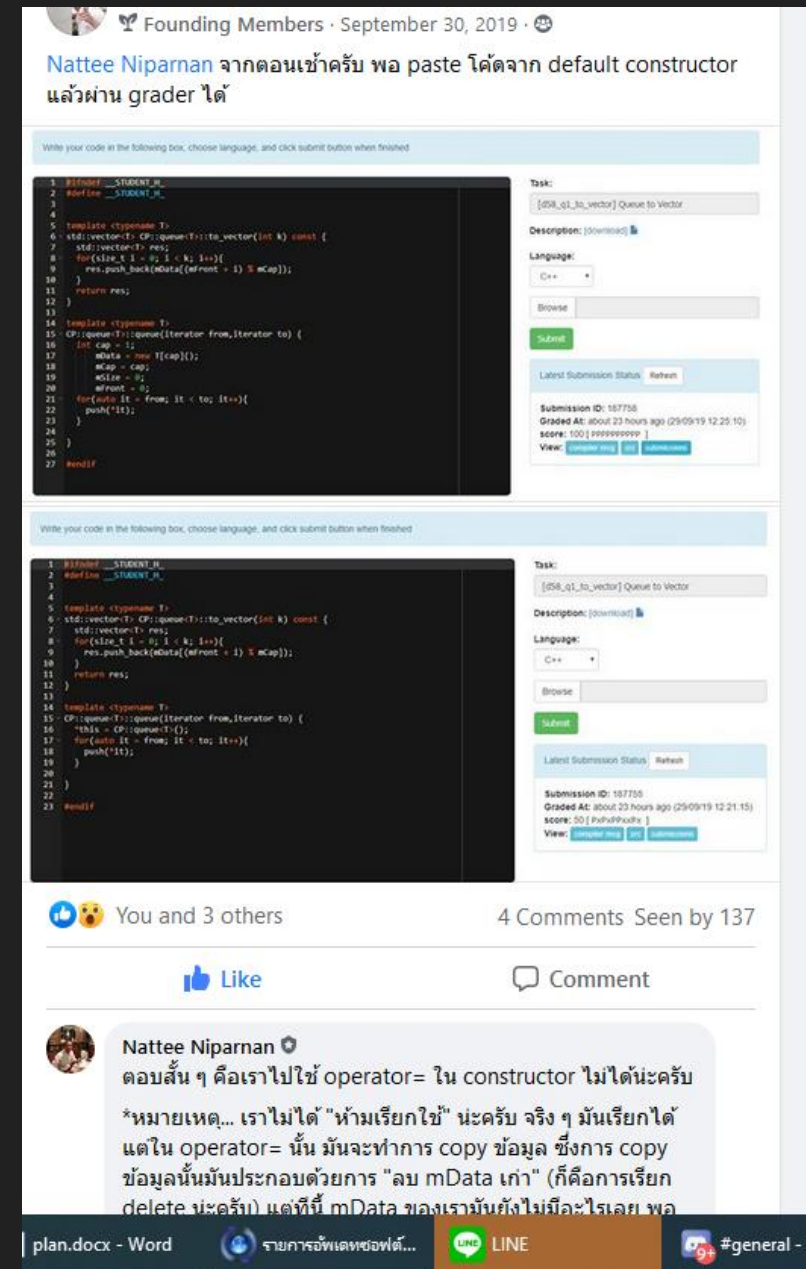
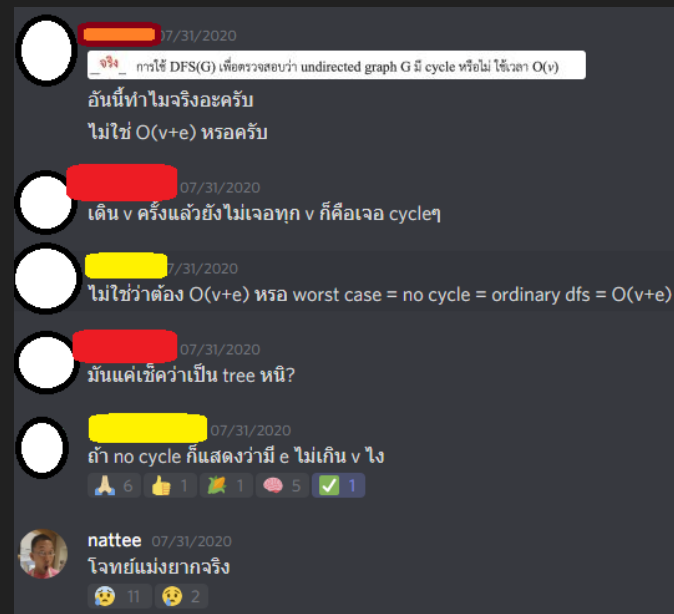
- `Travel(vector<pair<string,vector<int>>> &hotel, vector<pair<string,vector<int>>> flight,int w)` เป็น constructor ซึ่งรับข้อมูล hotels และ flights ซึ่งระบุว่าโรงแรมแต่ละโรงแรม และ เที่ยวบินแต่ละเที่ยวบินรองรับพื้นที่ใดบ้าง โดยให้ `hotels[i].first` คือชื่อของโรงแรม และ `hotels[i].second` คือรายการของพื้นที่ที่โรงแรมนั้นรองรับ และ `flights[i].first` คือชื่อของเที่ยวบิน และ `flights[i].second` คือรายการของพื้นที่ที่เที่ยวบินนั้นรองรับ และ `w` คือจำนวนวันเข้าพักซึ่งมีเลขประจำตัวตามที่ระบุในตัวแปร tourist
- `void add_stay(string hotel, int date, vector<int> tourist)` เป็นฟังก์ชันที่ใช้ระบุว่าโรงแรมชื่อ hotel ในวันที่ date นั้นมีผู้เข้าพักซึ่งมีเลขประจำตัวตามที่ระบุในตัวแปร tourist
- `void add_passenger(string flight, int date, vector<int> tourist)` เป็นฟังก์ชันที่ใช้ระบุว่าเที่ยวบินรหัส flight ในวันที่ date นั้นมีผู้เดินทางซึ่งมีเลขประจำตัวตามที่ระบุในตัวแปร tourist
- `bool can_get_refund(int id, string flight, int date)` เป็นฟังก์ชันสำหรับตรวจสอบว่าคนที่ มีเลขประจำตัวเป็น id นั้นสามารถขอคืนเงินค่าตัวสำหรับเที่ยวบิน flight ในวันที่ date ได้หรือไม่
  - ฟังก์ชันนี้อาจจะถูกเรียกโดยค่า id, flight หรือ date ใด ๆ ก็ได้ รวมถึงค่าที่ไม่เคยพบมาก่อนในระบบก็เป็นได้
  - ฟังก์ชันจะต้องคืนค่า true ก็ต่อเมื่อ นักท่องเที่ยวสามารถได้รับเงินคืนตามกฎหมายข้างต้นเท่านั้น
  - ให้คืนค่า false ในกรณีอื่น ๆ ทั้งหมด

จงตอบคำถามต่อไปนี้



# งตรงไหนก็มาถาม

- ถามส่วนตัวก็ได้
- ถาม broadcast ก็ได้ (เชียร์แบบนี้ เพราะเพื่อนได้ประโยชน์ด้วย)
- เข้าใจว่าเขิน แต่ทำได้ก็ดีครับ





# ช่องทางการติดต่อสื่อสาร

- ปรึกษา
  - Discord CP48 #discuss-data
- ประกาศทางการ
  - myCourseVille

# ยังลงทะเบียนไม่ได้ ทำอย่างไรดี

- นิสิตภาคคอม ทุกคนต้องได้เรียน
- นิสิตภาคอื่น จำกัดจำนวน และ อนุญาตเฉพาะคนที่ต้องการย้ายเข้าภาคคอมเท่านั้น
  - ต้องผ่านเงื่อนไขของการย้ายภาคของภาคคอมทั้งหมด
    - Comp Prog ได้ A ขึ้นไป
    - เกรดเฉลี่ย 3 ขึ้นไป
- คนที่ไม่ผ่านเงื่อนไขดังกล่าว กรุณาลดรายวิชา
  - มิเช่นนั้นจะกลายเป็น W
- เดี่ยวจะมีช่องทางให้แจ้งจำนวน
  - ต้องรอเพิ่มจำนวนใน reg.chula แล้วไปเพิ่มเอง

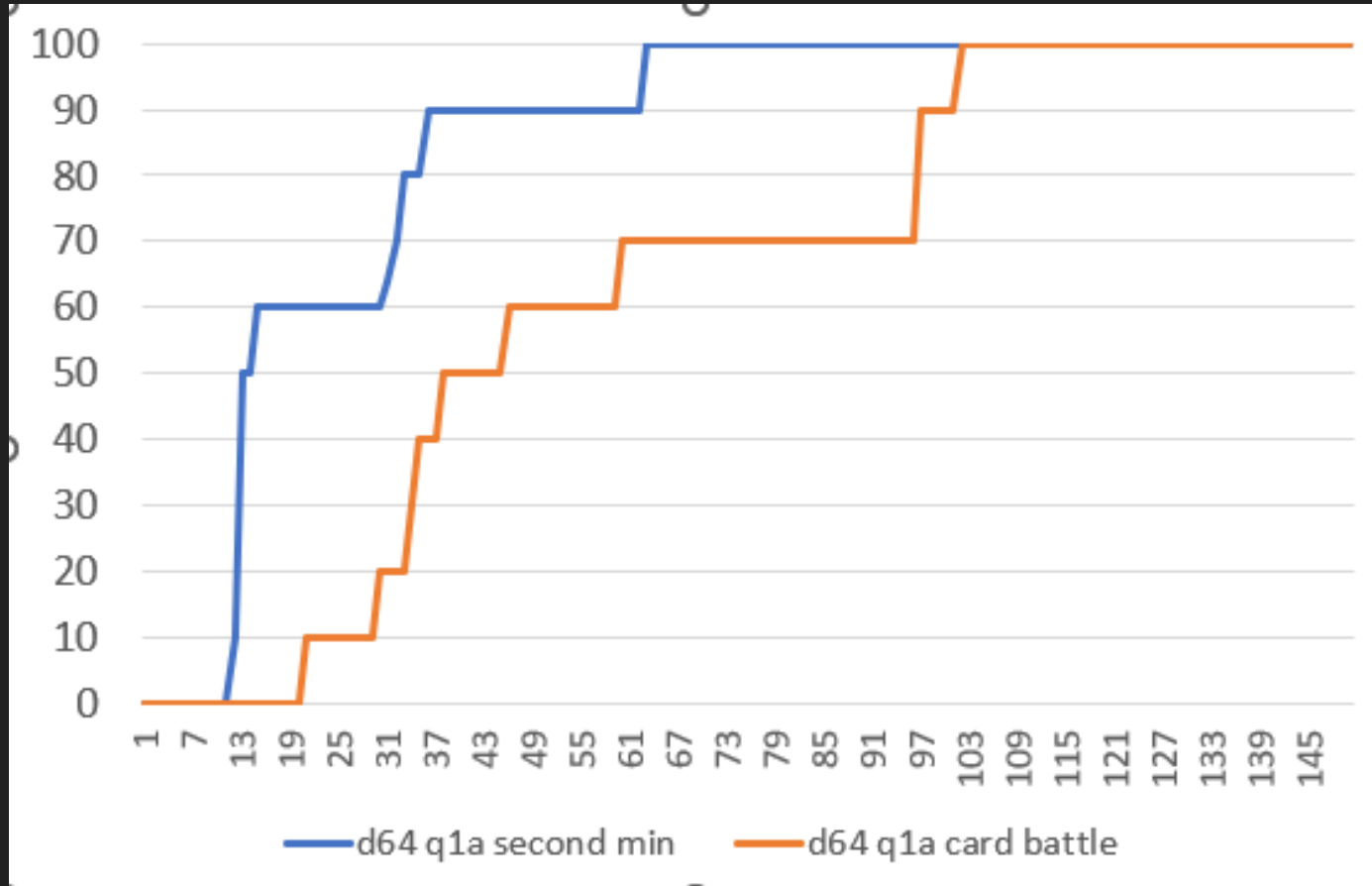
# คะแนน

- Midterm 25% (Paper Based)
- Final 30% (Paper Based)
- Quiz 40%
  - รวบรวม ๆ เดือนละครั้ง
  - ใน ชม. เรียน
- Unknown 5% (might be converted to quiz)

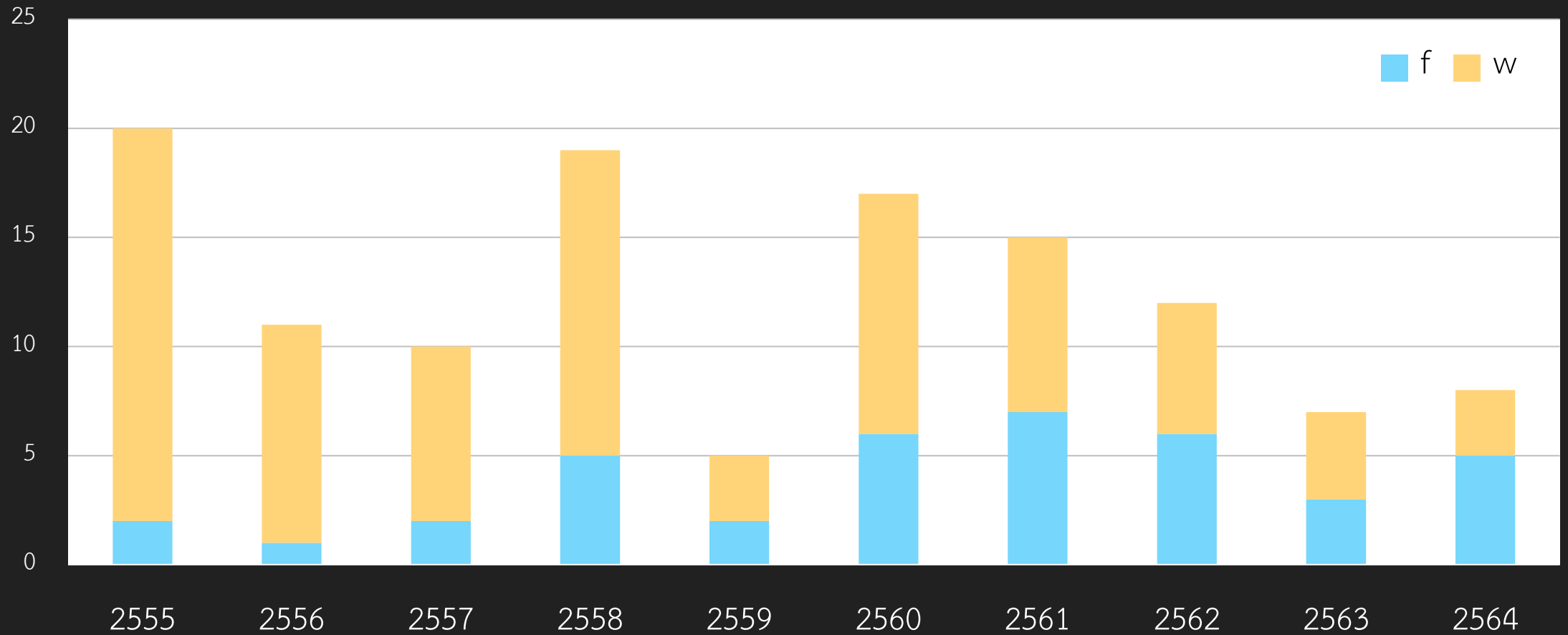
# Warning

- วิชานี้ไม่ยาก แต่ต้องอาศัยการฝึกซ้อม
- อย่าทำอะไรวันสุดท้าย!
  - สอบวันที่  $x$  ควรจะเตรียมตัวตั้งแต่วันที่  $x-1$

# Quiz Historical Data



# F & W Historical Data



# กฎในการเรียน

1) ไม่จำเป็นต้องชุดนิสิต แต่ขอให้สุภาพ

- ห้ามขาสั้น ห้ามรองเท้าแตะ
- ยืนสั้ได้ เสื้อ shop ก็ได้
- ขอย้ำว่าผิดกฎหมายมหาวิทยาลัย (ผู้สอนอนุญาตเฉพาะในวิชานี้ ส่วนวิชาอื่น ๆ หรือบริการอื่น ๆ ไม่เกี่ยวข้องกัน)
- ตอนสอบ “ต้อง” แต่งกายชุดนิสิต

2) ใส่หน้ากากอนามัยตอนมาเรียน

3) ไม่มีคะแนนเช็คชื่อ อยากเข้าก็เข้า ไม่อยากเข้าก็ไม่ต้อง

(ยกเว้นตอนสอบ)

# Link สำคัญ ๆ

- เดี่ยวแปะไว้ให้ใน mCV ด้วย

- video

<https://www.youtube.com/playlist?list=PLW3DcQsnGanPGhY2Y0A9hc45KnfS55RZI>

- source code หลักอยู่ที่ <https://github.com/nattee/data-class>
- grader ของวิชานี้ อยู่ที่ <https://nattee.net/grader>
- ข้อสอบเก่าอยู่ที่ <https://nattee.net/teaching>



# I have a dream

- ต้องเคยพบกับ ความสนุก ในการ แก้ปัญหา
- จบวิชานี้แล้วต้อง เก่งขึ้น!
- ไม่กลัวการเรียนรู้สิ่งใหม่

“จุดหมายปลายทางที่แท้จริงของการศึกษา ก็  
เพื่อจะฝึกจิตใจให้มีระเบียบวินัย  
ไม่ใช่แต่เพียงจะสะสมความรู้ หรือวิธีการต่าง ๆ  
และเพื่อฝึกให้ใช้กำลังความคิดของตนเอง  
มากกว่าที่จะอาศัยความคิดเห็นของผู้อื่น”

พระเจริญวิศวกรรม