Topic 0

Class Introduction

資料結構與程式設計
Data Structure and Programming

Sep, 2012

Class Information

◆ Class Website
- https://ceiba.ntu.edu.tw/1011dsnp

◆ Discussion board
- PTT → EE_DSnP
- FB → ??? (TBD)

◆ My office:
- EE building II - 444
- (Tel) 3366-3644
- (e-mail) ric@cc.ee.ntu.edu.tw
- Office hour: stop by or by e-mail appointment

◆ Class TA(s)
- TBD

Class Information

- Required textbook: none
- Suggested reading
  - Class slides and source codes
    - Download from the Ceiba website
  - Any of your Data Structure and C++ programming textbooks
- Highly recommended (DO THEM ASAP)
  - Review C++
  - Get access to and familiar with Linux workstations

Grading (May subject to change)

Homework	60%
Final project	40%
Bonus	TBD

The final grades are subject to linear adjustment. Instructor will determine the average and standard deviation

What is this class about?

- ◆ People say that this class is more about programming (P), and less on data structure (DS).
- ◆ Indeed, I intend to use DS as a vehicle to teach you how to write a good program.
- ◆ However, to write a good program, you must cleverly utilize DS, and even define your own DS.
  - ● So, DS + P is a good combination to learn P.
  - ● You are encouraged to take other course in EE or CS department if you want to learn more about DS.

Before I get into detailed course introduction, let me clearly state some principles and expectations so that you can decide whether you want to stay or leave.

聽說這門課很操，是真的嗎?

◆不要懷疑，根據多次的問卷統計，同學們覺得這門課的 loading 大約 >= 9 學分，每兩個星期要花 20 ~ 30 hours (以上) 在作業上。

◆但好處是沒有期中 & 期末考，不用去 K 教科書或是消習題。

- 不過有期末 project
- 而且要學會自己找參考資料

◆所以如果你還要忙社團或是要參加什麼隊的，或是其他的課很重，請搞清楚你的 availability，切莫始亂終棄!!

我是個寫程式的小嫩咖，我有辦法修這門課嗎?

◆原則上絕大部分的人在你們這個年紀都是寫程式的小嫩咖，所以我想沒有問題。

◆重點還是要能有 “commitment”

- 再強調一次，要考量現實，不要輕易相信自己的意志力可以戰勝一切!

◆Commitment 從何而來?

- 首先，請確定 “把程式學好” 對你的重要性
- 再來，請確定自己可以接受 “學習比成績重要”
- 還有，請發誓自己 “寧願被當，也不會抄襲”

關於抄襲

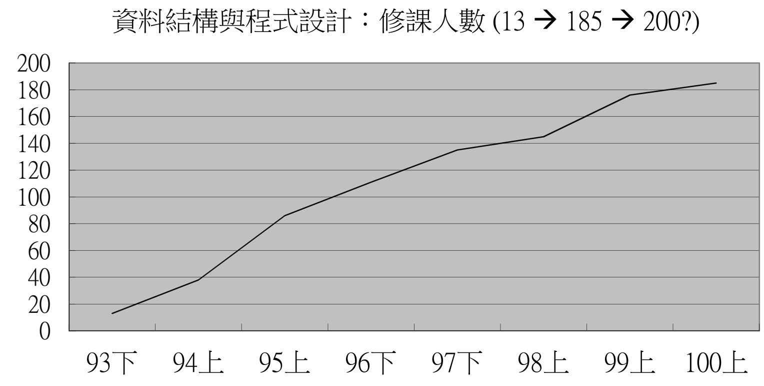
- ◆ 我們有強大的抓抄襲的程式，所以請勿抱著苟且的想法。
- ◆ 歡迎互相討論，甚至拿別人的 code 來 study 也不會/無法禁止 (雖然這樣並不好)，但最後一定要自己寫。
- ◆ 抓抄襲程式會對所有的作業以及之前學長姊的作業去做比對，如果沒有抄襲，相似度都會很低，但如果有抄襲，不管你是改變數名稱， 還是換 statements 順序... 等等，我們都可以很容易抓出來
  - ● 以我們的作業複雜度而言，只要是自己寫的，一定一眼就可以看出跟抄襲的不同。
- ◆ 過去: 規定抄襲者一律學期成績 0 分 ➔ 心軟而沒有確實執行
- ◆ 今年: 凡抄襲者不論多寡、理由，除該次作業 0 分之外，學期成績一律再扣 20 分 (調分後)

為什麼是 C++? 為什麼不直接學 APP 就好?

- ◆ 我們有許多優秀的 CS 人才，但卻沒有一個像樣的 CS 產業，WHY?
- ◆ PC 時代，“軟體” 為主要獲利之 “商品”
  - ● 我們靠生產 Intel CPU 周邊的IC 與附件成就了95 ~ 05 的高科技奇蹟
- ◆ 後 PC 時代，”廣告“、”服務” 為主要的獲利
  - ● Yahoo, Google, Amazon, e-Bay, FaceBook 的崛起，我們的定位在哪裡?
  - ● APP 是個產業嗎? What should we do?
- ◆ 大部分的網路程式、APP 等等，技術門檻其實都不高，但要寫得快又好，而且能夠 “長大”，需要的就是良好的寫程式的 “sense”，以及堅持的 “原則”，還有正確的 “optimization” 的概念，這些都必須從比較低階、複雜的語言 (即 C++) 來學，才會學得透徹。

有教無類? 教學品質?

◆Well, as you can see, the class is overbooked.



Should I stay or should I go?

◆After taking the class, somebody liked it, but somebody hated it.

◆去年 185 個選課的同學，最後 18 個停修

- 電機大二: 3/32
- 電機大三: 12/110
- 電機大四: 0/31
- 資工: 0/2
- 外系 (物理、心理、數學、機械、工科、資管): 2/8
- 研究所 (電信、藥理): 1/2

Some statistics about the grades

	100-1	99-1	98-1
外系	80 (6/8)	82 (5/7)	61 (5/5)
電二	83 (29/32)	83 (27/27)	--- (0/1)
電三	79 (98/110)	84 (101/106)	88 (74/83)
電四	82 (31/31)	86 (17/22)	82 (27/38)
資工	92 (2/2)	84 (7/11)	87 (11/15)
研究所	63 (1/2)	52 (2/3)	58 (3/3)

“Should I stay or should I go?”

◆Please check on your own:

1. Do I have the eager to improve my programming skill?
   - 光有 “希望” 是不夠的，要有 “渴望” 才行。
2. Am I willing to spend more than 10 hours per week on the homework?
   - 獨力完成，不抄襲，也不要當寄生蟲。
3. Do I agree that “learning” is the most important thing in class?
   - 心態上要能接受 “學習” 比 “分數” 重要。

FAQs & Suggestions

- Can I take this class as I am not an NTUEE student?
  - You are also welcome, but you are advised to find someone to study and discuss together.
- Can I sit in this class?
  - Well, technically there is no restriction on sitting-in.
  - However, since the number of students is way too high, please leave the seats to the students who take this class.
- Is this the last time I offer this class?
  - Nobody knows. But I will try to sign in this class as long as it is possible.
  - Please note that other professors also offer this class in different semesters.
- My only request to you: 做人要甘願!!
  - If you decide to stay in this class, you need to know that this is a heavy class.
  - Don’t blame on me if you find it too heavy-loaded!

歡喜修課，甘願承受

- 說實在的，DSnP 是 NTU(EE) 的奇蹟!
  - 需要大家共同的珍惜
- 非誠勿試，please!!

Some last words...

◆如果人太多的話，我們會和隔壁連線 (MD 205)
- 旁聽生，以及找不到插座使用的同學請到隔壁，謝謝!!

◆[希望] C++ review 會多放點例子

◆[希望] 多留一點時間講解 homework

◆[希望] 能給 Homework #6 的 solution code, 免得 final project 會太硬

◆請多多利用 PTT/EE_DSnP 討論問題!!

Course Outline

Part 1: Introduction

0. Class Introduction
1. Data Structure in Programming

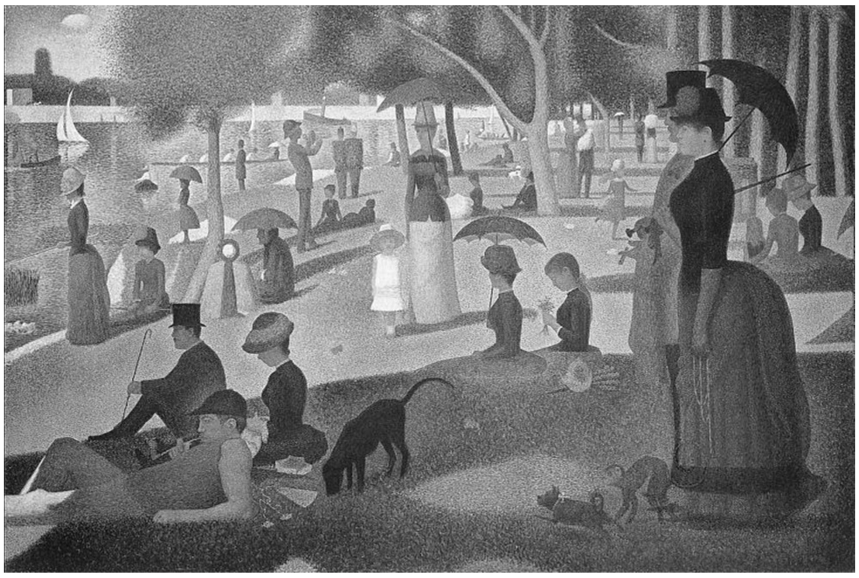
Why is data structure (implementation) so important?

1. Data Structure in Programming: *Why is data structure (implementation) so important?*

- Why do you learn DS?
  - When will you use it in your daily life? If you don't apply it in your programs…?
- "Programming is an art; DS is the spirit of the art."
  - If you know how to cleverly utilize DS in your codes, you will definitely produce an elegant program.
  - Masterpiece? 99% perseverance and 1% talent
- "Writing program is an ego thing,

  while writing a SW tool/framework needs

  cooperation"



Georges Seurat, *"A Sunday Afternoon on the Island of La Grande Jatte"*, 1884-1886

Data Structure in Programming

- As we will see, “programming” is nothing more than “storing” and “operating” data.
- “Data structure”, in general, includes all types of “structured storage” in which data can be “operated” in various ways.
- Object oriented programming (OOP) teaches you how to use “structured data type” (e.g. *class*) to write a good program.

How to be a good programmer?

- My observation
  - Achievements in ACM or programming contests do NOT necessarily imply good programming skill.
  - It just means that you are smart, or at most, good in math and logic.
- Our objective here is not just to be a good programmer, but a good program **designer**.
  - Has the capability to plan, architect, and manage a large scaled program.

Course Outline

Part 1: Introduction

0. Class Introduction
1. Data Structure in Programming

   Why is data structure (implementation) so important?

2. Programming on Linux Workstations

   A peek in the real engineering world

2. Programming on Linux Workstations: *peek in the real engineering world*

- Why Linux? Why not M$ Windows?
- History of Linux OS
- Basic survival guide on Linux
- Writing programs on Linux
  - Shell commands
  - Compiler
  - Makefile
  - Debugger

Homework #1.1

- Target due date: Week 4 (10/03)
- You MUST have access to Linux to do this homework
  - Install Linux on virtual machine (e.g. VirtualBox, VMware)
  - Has an account on some Linux workstation (e.g. PC room, your lab)
  - Dual boot your computer

1. Understand your Linux environment
2. Shell script
3. A simple makefile

Overview of this course

Part 1: Introduction

Part 2: Polishing Your Programming Skills

Part 3: Data Structure Revisited

Part 4: Putting What You Learn Together

3. C++ Advanced Features Review: *When can/should I use them?*

- ~~Object, pointer, reference~~
- ~~Const, static, extern, type cast~~
- ~~Namespace~~
- ~~Constructor, destructor~~
- ~~#include, #define, #ifdef~~
- ~~Enum, union, bit slicing~~
- ~~Public, private, friend~~
- ~~Inheritance, virtual, polymorphism~~
- ~~Operator overload~~
- ~~Template~~
- ~~Functional object~~
- ~~Stream classes~~
- ~~String~~
- ~~Exception handling~~

3. C++ Advanced Features Review: *When can/should I use them?*

- Understanding "variables"
  - Object, pointer, reference
  - Const, static, extern, type cast
  - #define, typedef
  - Namespace
- Understanding "classes"
  - Constructor, destructor
  - Enum, union, bit slicing
  - Public, private, friend

3. C++ Advanced Features Review: When can/should I use them?

- Understanding "overloading"
  - Function & operator overloading
  - Function & class template
- Understanding "polymorphism"
  - Class inheritance, virtual function
  - Functional object
- Understanding "libraries"
  - #include, #ifdef
  - Stream classes
  - String
- Exception handling

Homework #1.2 and #2

- Homework #1.2 (target due: 10/10)
  - C++ advanced feature practice (overloading, template, polymorphism)
  - Homework assignment will be announced before the lectures on these topics.
- Homework #2 (target due: 10/17)
  - A command line reader
  - Thorough understanding of "pointers"
  - Basic program design
  - Ref code: 627/708 lines C++ (last year's)
  - New feature(s) may be added...

A short version of "Computer Programming" class?

- NO!!
- If you don't have any background in C++ (or C) ...
  - You probably have chosen the wrong class.
- If you are poor in C++ programming...
  - Well, you are definitely NOT the only one, so you are very welcome!!
  - Please pay attention to the lectures in this topic, and make sure you can commit enough time on homework

You may think I cover way too many details in C++... (Why bother to understand them?)

- Remember:
  Programming is a computer science.
  - There is NO random bug!!
    Everything happens for a reason.
  - You need to be rationale, and be "precise on the details".
  - ➔ Capability to handle the complexity!!
- But...
  Programming is also an art.
  - A good program looks beautiful!!
  - A beautiful program is beautiful for a reason.
  - A good design is a MUST, and easy to maintain to make the program live long!
  - ➔ Sense to manage the complexity!!

Course Outline

Part 2: Polishing Your Programming Skills

3. C++ Advanced Features Review:
   When can/should I use them?
4. STL Basics:
   The Standard Template Libraries
5. What is a Good Program?
   Software engineering point of view
6. Memory Management:
   How to gain 30% performance improvement easily

4. STL Basics:
The Standard Template Libraries

- Why template libraries?
- Why standard?
- The standard template libraries
  1. Container classes
     - List, array, map, hash, stack, string, bitvector, etc…
  2. Iterators
     - Forward, bidirectional, random, etc
  3. Algorithms
     - For_each, sort, partial_sum, sort, etc.
  4. Functional object
     - Unary, binary, arithmetic, etc
  5. Utility
  6. Memory allocation

5. What is a Good Program?
software engineering point of view

- What do you suffer most in programming?
  - Coding? Compiling? Debugging?
- Which one is more important?
  - Best or complete algorithm?
  - Least instructions/sub-routines called?
  - Least memory used?
  - Smaller size of code?
  - More (or less) advanced language features?
  - Easier to debug and maintain?
  - Nicely documented?
  - Easily reusable?
- Coding style guideline

6. Computational Complexity:
Time and space tradeoffs

- Review of complexity analysis
- Why should I care?
- What’s the most frequently encountered problem?
- What’s your best bet?

7. Memory Management: *How to gain 30% performance improvement easily*

- Where's your bug?
  - Segmentation fault, bus error, etc
- Constructor and destructor
- Fragmentation
- System memory allocation/deletion
- Implement your own memory manager
- Garbage collection
- Cache effect

Homework #3 & #4

- Homework #3 (target due: 10/31)
  - Complete command interface and a simple command-line modular calculator.
  - Learn how to write a structured code
  - Ref code: 1541(1814)/2015 lines C++
- Homework #4 (target due: 11/14)
  - Memory management
  - Pointers (again), basic data structure
  - Ref code: 1328(2334)/2520 lines C++

Overview of this course

Part 1: Introduction

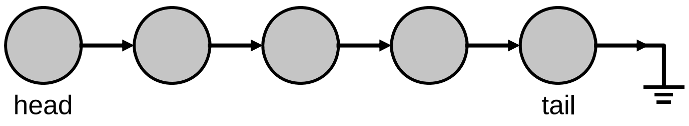
Part 2: Polishing Your Programming Skills

Part 3: Data Structure Revisited

Part 4: Putting What You Learn Together

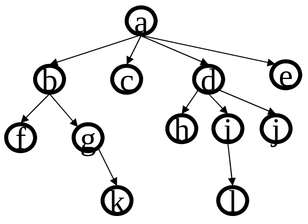
8. Dynamic Array vs. Linked List: *Which one is better?*

- Linear data types
- Static vs. dynamic array
- Why dynamic array? Why not linked list?
- How to evaluate their performance?
  - Runtime vs. memory usage



9. Tree: *How to search data faster than linear time?*

- ◆Non-linear data types
- ◆Decision trees
- ◆Tree traversal
- ◆Balanced trees
- ◆Implementation issues

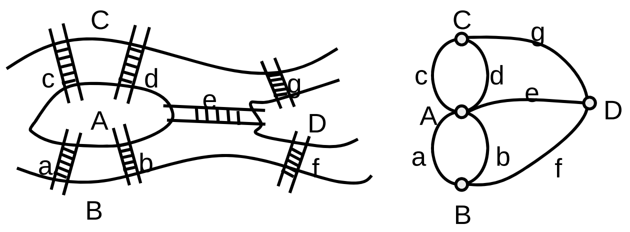


Homework #5

- ◆Target due: 11/28
- ◆Implementation and comparison of various data structures
  - ● Linked list
  - ● Dynamic array
  - ● Binary search tree
- ◆Ref code: 1268(2274)/3062 lines in C++

10. Graph and Circuit:
From CS to EE applications

- Tree vs. graph
- Basic graph theories
- Graph traversal problems
- Loop handling
- How to design data structure for a circuit netlist?

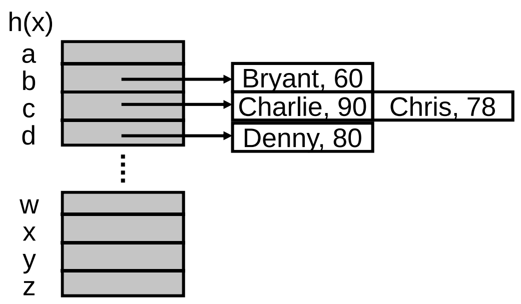


11. Heap, Set and Map:
How to store sorted data?

- Review of sorting algorithms
- Review of binary (balanced) trees
- Complexity analysis
- Alternative ways of implementation
- Standard Template Library (STL) revisit

12. Cache vs. hash: *Virtual memory in your program*

- Review on hash
- Alternative to hash
- What's the difference?
- Computational cache/hash



Homework #6

- Target due: 12/12
- A circuit parser
  - I/O and file streams
  - Graph/Circuit data structure
  - Hash/Map usage
  - Boolean logic
- Ref code: 1482(2736)/4311 lines in C++
- A special lecture note on "Lex and Yacc" may be offered

13. Bit Vector and Matrix: *All about numerical operations*

- Bitwise operations
- Beyond 32/64 bits
- Multi-valued system
- Dense vs. sparse matrix
- Matrix operations
- Linear algebra…

Overview of this course

Part 1: Introduction

Part 2: Polishing Your Programming Skills

Part 3: Data Structure Revisited

Part 4: Putting What You Learn Together

Final Project

- ◆ Functionally Reduced And-Inverter Graph (FRAIG)
  - Read in a circuit netlist (HW6)
  - Perform circuit optimization (graph operations)
  - Simulate the circuit (graph traversal, Boolean operations)
  - Collect functionally equivalent candidate pairs (efficient hash implementation)
  - Define the "magic number" to control the program flow (engineering sense)
- ◆ Ref code: 4275(5281)/7242 lines in C++
- ◆ 40% of the final grade!! Please start earlier!!

Overview of this course

Part 1: Introduction

Part 2: Polishing Your Programming Skills

Part 3: Data Structure Revisited

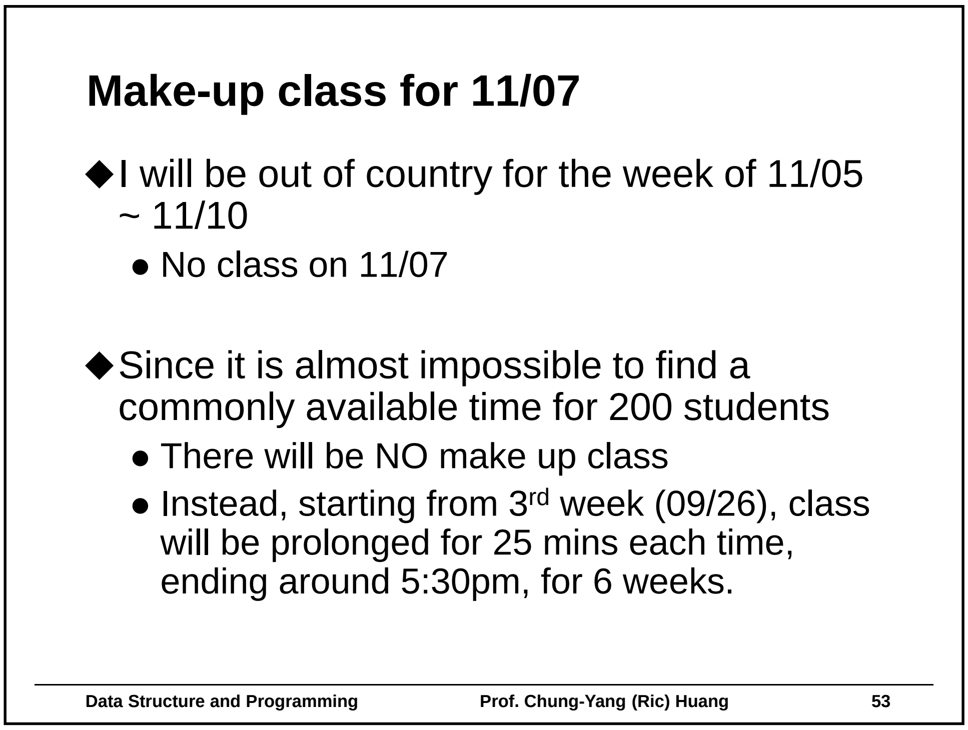
Part 4: Putting What You Learn Together

Class Schedule

09/12	Class Intro, DS in Prog.		
09/19	Linux Prog., C++ Review	HW1.1 out	
09/26	C++ Review	HW1.2 out	
10/03	C++ Review	HW2 out	HW1.1 due
10/10	National Holiday (no class)		HW1.2 due
10/17	C++ Review	HW3 out	HW2 due
10/24	STL,Good Prog., Complexity		
10/31	Complexity, Mem Mgr	HW4 out	HW3 due
*11/07	**ICCAD (no class)**		

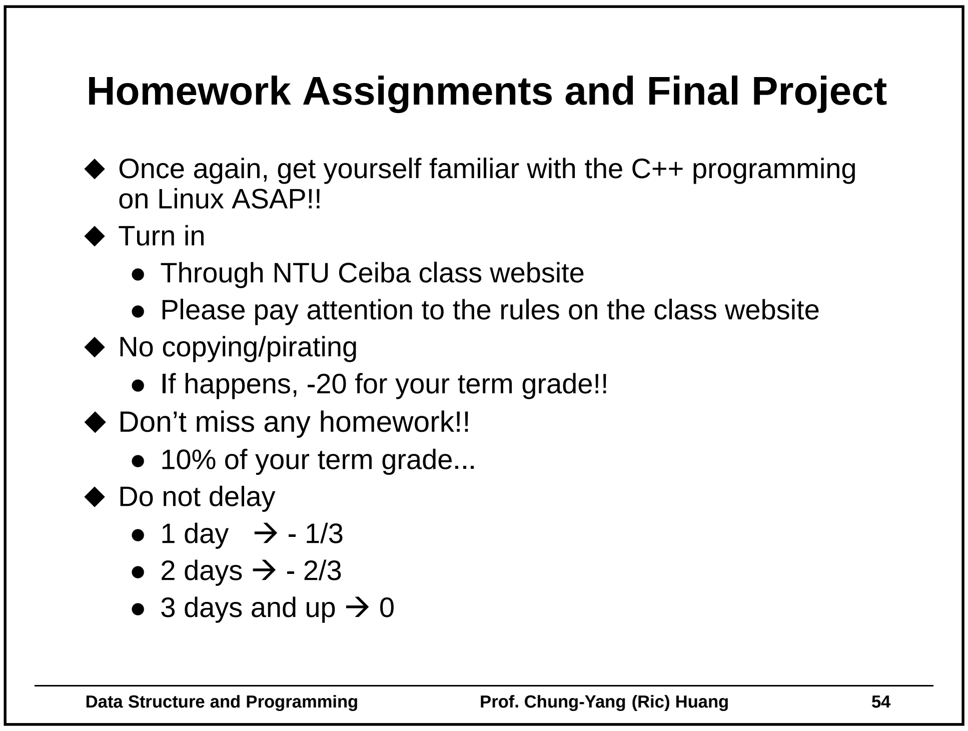
Class Schedule

11/14	Array and List, Tree	HW5 out	HW4 due
11/21	Tree, C++ Review		
11/28	C++ Review, Graph	HW6 out	HW5 due
12/05	Graph, Heap, set, Map		
12/12	Cache and Hash	Proj. out	HW6 due
12/19	Final Project Discussion		
12/26	Final Project Discussion		
01/02	Bit Vector and Matrix		
01/09	Final exam week		
01/16	Final proejct week		Proj. due



Make-up class for 11/07

- ◆I will be out of country for the week of 11/05 ~ 11/10
  - No class on 11/07
- ◆Since it is almost impossible to find a commonly available time for 200 students
  - There will be NO make up class
  - Instead, starting from 3rd week (09/26), class will be prolonged for 25 mins each time, ending around 5:30pm, for 6 weeks.



Homework Assignments and Final Project

- ◆ Once again, get yourself familiar with the C++ programming on Linux ASAP!!
- ◆ Turn in
  - Through NTU Ceiba class website
  - Please pay attention to the rules on the class website
- ◆ No copying/pirating
  - If happens, -20 for your term grade!!
- ◆ Don’t miss any homework!!
  - 10% of your term grade...
- ◆ Do not delay
  - 1 day → - 1/3
  - 2 days → - 2/3
  - 3 days and up → 0