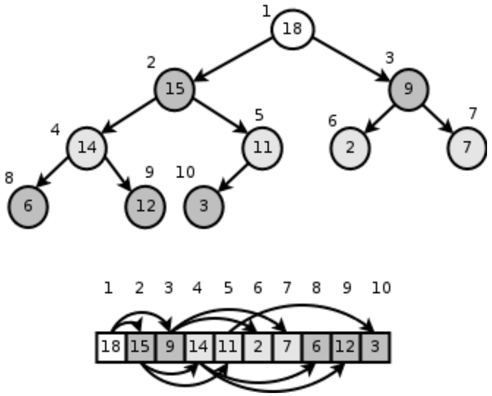
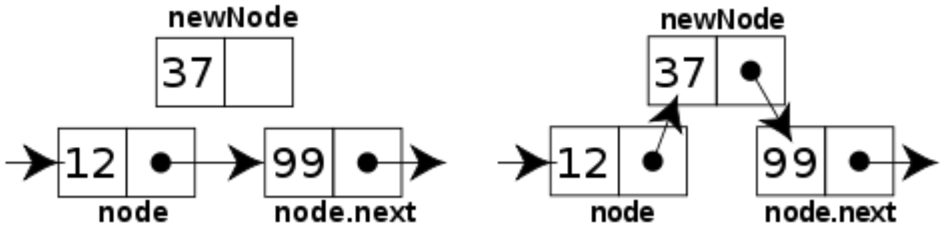


```
INSERTION-SORT(A)                                   cost   times
for j = 2 to A.length                               $c_1$    $n$
    key = A[j]                                      $c_2$    $n-1$
    // Insert A[j] into the sorted
            sequence A[1..j-1].                     0      $n-1$
    i = j - 1                                       $c_4$    $n-1$
    while i > 0 and A[i] > key                      $c_5$    $\sum_{j=2}^{n} t_j$
        A[i + 1] = A[i]                             $c_6$    $\sum_{j=2}^{n} (t_j - 1)$
        i = i - 1                                   $c_7$    $\sum_{j=2}^{n} (t_j - 1)$
    A[i + 1] = key                                  $c_8$    $n-1$
```



WELCOME TO CS 24!

Problem Solving with Computers-II

Instructor: Diba Mirza

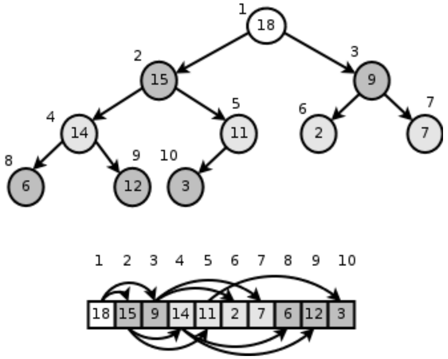
```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;

}
```

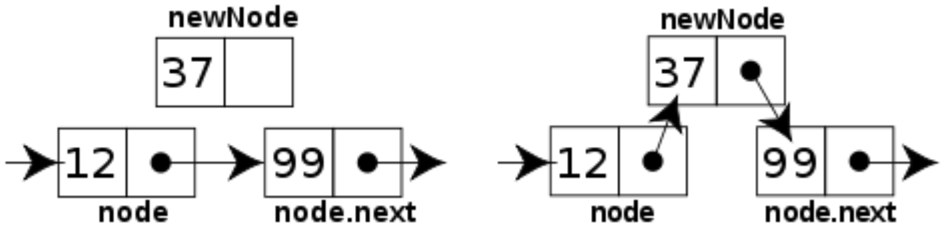
Read the syllabus. Know what's required. Know how to get help.

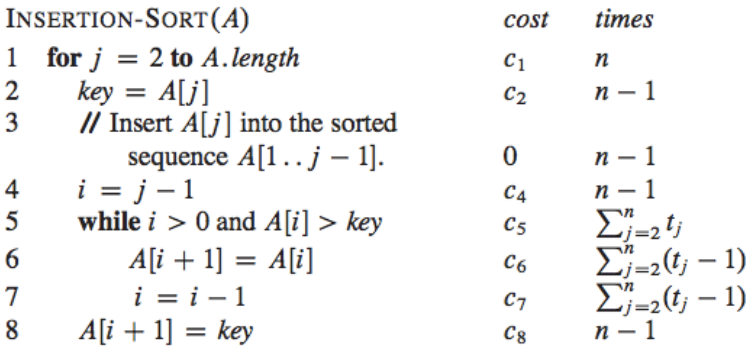
About this course

You will learn to:

- Design and implement **larger programs** that **run fast**
- Organize **data** in programs using **data structures**
- **Analyze** the **complexity** of your programs
- Understand what goes on **under the hood of programs**







INSERTION-SORT(A)		*cost*	*times*
1	**for** $j = 2$ **to** $A.length$	c_1	n
2	$key = A[j]$	c_2	$n-1$
3	// Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4	$i = j - 1$	c_4	$n-1$
5	**while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6	$A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n}(t_j - 1)$
7	$i = i - 1$	c_7	$\sum_{j=2}^{n}(t_j - 1)$
8	$A[i+1] = key$	c_8	$n-1$

Data Structures and C++ **Complexity Analysis**

About the team

Instructor: Diba Mirza

- **TAs:** Lucas Nguyen, Ganesh Sankaran, Roman Aguilera
- **ULAs:** Tina Shi and Zack Glazewski

- Communication with staff via **Piazza**
- Lectures, sections, OH will be remote for the first two weeks
- Include [CS24] in the subject line of any email communication with me

Note: OH schedule may change after we switch to in person

** Ask questions about class examples, assignment questions, or other CS topics. **

Course Logistics

- Coure website: https://ucsb-cs24.github.io/s22
- If you have a section conflict, you may informally switch your section time. Post to the “section swap” thread on Piazza to announce the switch.
- NO MAKEUPS ON EXAMS!
- Start assignments early and get a “timeliness” bonus!
- To complete the labs you need a college of engineering account. If you don’t have one yet, send an email to help@engineering.ucsb.edu

iClicker Cloud

- Instructions to register for iclicker cloud for free are on Gauchospace
- Download the iclicker REEF app to participate in class

Required textbook

Zybook: CMPSC 24: Problem Solving with Computers II

Recommended textbook

- Problem Solving with C++, Walter Savitch, Edition 9

You must attend class and lab sections
You must prepare for class
You must participate in class

About you…

What is your familiarity/confidence with C++ memory-management (stack vs heap)?

A. Know nothing or almost nothing about it.

B. Used it a little, beginner level.

C. Some expertise, lots of gaps though.

D. Lots of expertise, a few gaps.

E. Know too much; I have no life.

About you…

What is your familiarity/confidence with using git version control ?

A. Know nothing or almost nothing about it.

B. Used it a little, beginner level.

C. Some expertise, lots of gaps though.

D. Lots of expertise, a few gaps.

E. Know too much; I have no life.

About you…

Have you implemented a linked list before in any programming language?

A. Yes

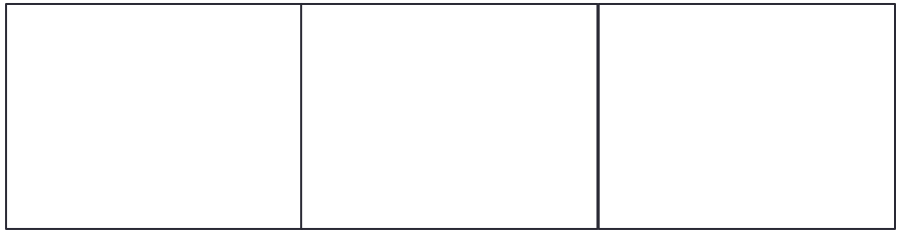
B. No

About lectures

- I will not be a talking textbook
- I love interaction: Ask questions anytime!
- I’ll ask you questions too! Be ready to discuss with the people near you and respond to multiple choice questions (using the clickers).
- Take a moment to introduce yourself to the people sitting near you.
  - Talk about your background and what you hope to get out of this class!

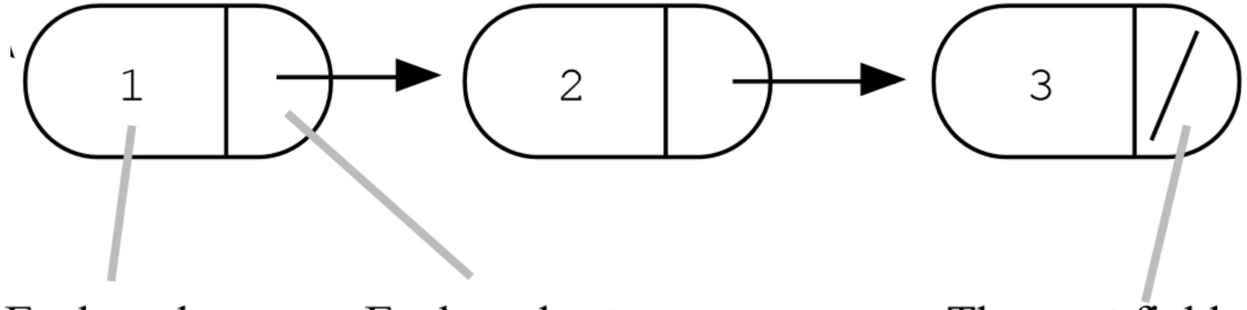
Linked list vs Array

Array



Defining the type Node

The overall list is built by connecting the nodes together by their next pointers. The nodes are all allocated in the heap.



Each node stores one data element (int in this example).

Each node stores one next pointer.

The next field of the last node is NULL.

Which of the following are valid ways of representing a linked list

A. `Node* head;`

B. `int* head = nullptr;`

C. `Node* head; Node* tail;`

D. Need to define a new type called `LinkedList`

```
struct Node {
     int data;
     Node *next;
};
```

Simplest Linked List (just a head pointer)

- Create an empty list
- Add a node with data “Tina Shi”

```
struct Node {
        string data;
        Node* next;
};
```

Next time

- Linked lists contd.