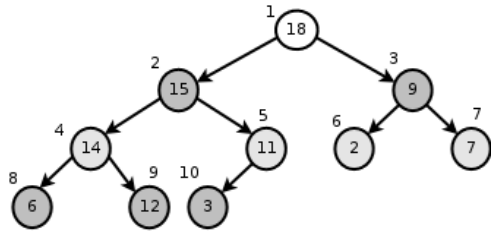
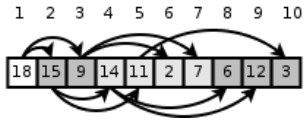
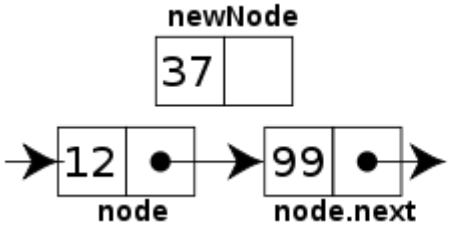
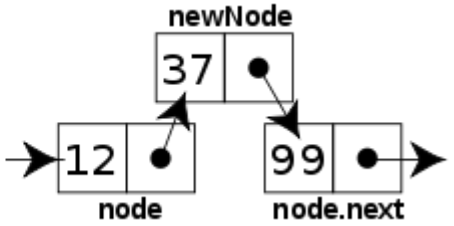




INSERTION-SORT(A)

		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	$n-1$





WELCOME TO CS 24!

Problem Solving with Computers-II

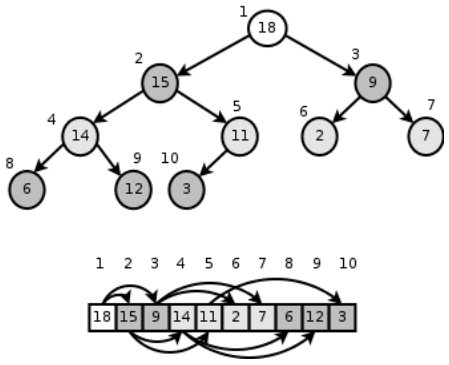
Instructor: Diba Mirza

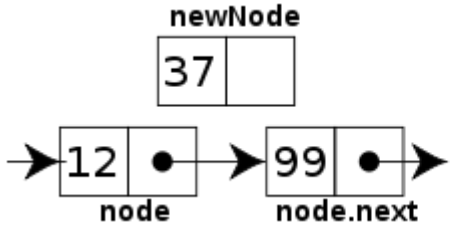
Read the syllabus. Know what's required. Know how to get help.

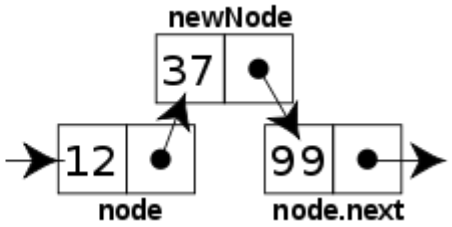
About this course

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 **C++** **Complexity Analysis**

Course Logistics

- Coure website: https://ucsb-cs24.github.io/s21/
- NO MAKEUPS ON QUIZZES and FINAL EXAM!
- Keep track of assignment due dates by reviewing the “weekly pattern” posted on Gauchospace
- No extensions on lab/programming assignments. Please plan to submit before the due date
- 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

About you…

What is your familiarity/confidence with Object Oriented Programming?

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 C++?

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 version control – git or subversion?

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 lectures

- I will not be a talking textbook.
- I love interaction: Ask questions anytime over chat but wait for a few minutes to get them answered.
- I'll ask you questions too! Be ready to discuss and participate over chat or by turning on your audio

- Practice: Ask me a question or share something interesting that happened over Spring break :)

Today's Learning Goals

- Integrate git command line into programming workflow.
  - Creating and cloning repos.
  - Git commands: git <status, log, add, commit, push>
- Review basics of classes
  - Defining classes and declaring objects
  - Access specifiers: private, public
  - Different ways of initializing objects and when to use each:
    - Default constructor
    - Parametrized constructor
    - Parameterized constructor with default values
    - Initializer lists
- Keep code organized: Write a simple Makefile

Git Demo

From lab00

Visit our [Github Sign Up Tool: https://ucsb-cs-github-linker.herokuapp.com/](https://ucsb-cs-github-linker.herokuapp.com/), login with your github.com account, click "Home", find this course (CS24-S21), and click the "join course button". That will automatically send you an invitation to join the course organization on github.
Log into GitHub to accept the invitation.

We will practice the following:

- Create a git repo in our class organization: ucsb-cs24-mirza-s21
- Clone the repo on your local computer or one of the CSIL machines
- Learn about what a git repo looks like in your file system (on a linux/unix/MAC) environment.
- Learn git commands:
  - git status
  - git add .
  - git commit
  - git log
  - git push

Concept: Classes describe objects

- Every object belongs to (is an instance of) a class
- An object may have fields, or variables
  - The class describes those fields
- An object may have methods
  - The class describes those methods
- A class is like a template, or cookie cutter

Concept: Classes are like Abstract Data Types

- An Abstract Data Type (ADT) bundles together:
  - some data, representing an object or "thing"
  - the operations on that data
- The operations defined by the ADT are the *only* operations permitted on its data
- ADT = classes + information hiding

```
class Dish{
public:
      void pourIn( double amount);
      void pourOut(double amount);
private:
      double capacity;
      double currentAmount;
};
```

Approximate Terminology

- instance = object
- field = instance variable
- method = function
- sending a message to an object = calling a function

Some advice on designing classes

- Always, *always* strive for a narrow interface
- Follow the principle of information hiding:
  - the caller should know as little as possible about how the method does its job
  - the method should know little or nothing about where or why it is being called
- Make as much as possible private
- Your class is responsible for it's own data; don't allow other classes to easily modify it!

What we have spoken about so far?

- Class = Data + Member Functions.
- Abstract Data Type = Class + information hiding
- How to activate member functions.
- But you still need to learn how to write the bodies of a class's methods.

Next time

- C++ Memory Model, Pointers and References