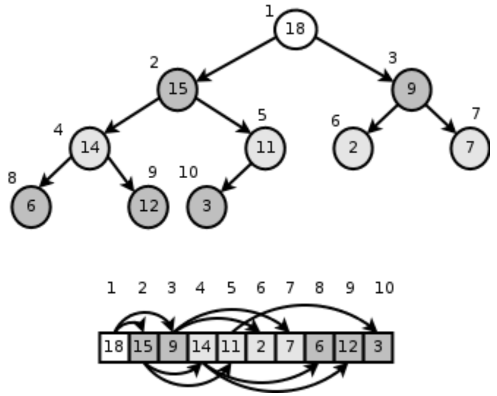
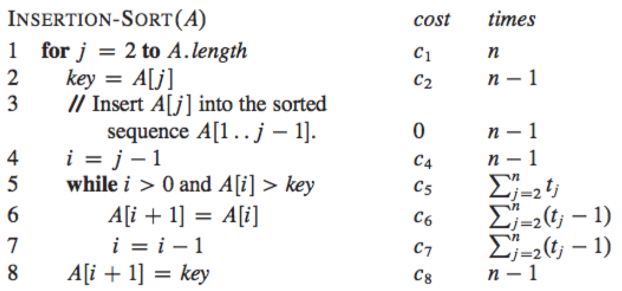
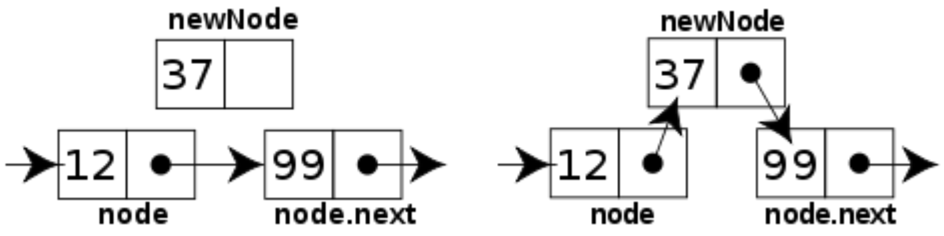




INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1\,..\,j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$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

```
#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.

Course website: https://ucsb-cs24.github.io/s25

About the team: we are here to support you. Use us!

- Prof. Mirza’s Office hours: Thurs 2p - 4p, HFH 1155, or by appointment
- Communication with staff via **Ed**
- Include [CS24] in the subject line of any email communication
- Sections start this week on Friday
- Office hours start next week

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

Ally (TA)

Brenna (TA)

Daniel (TA)

Kali (TA)

Sarah(TA)

Cindy (LA)

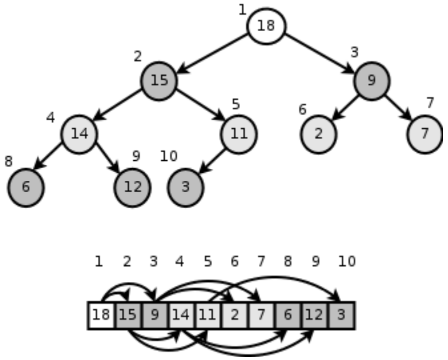
Nikhil (LA)

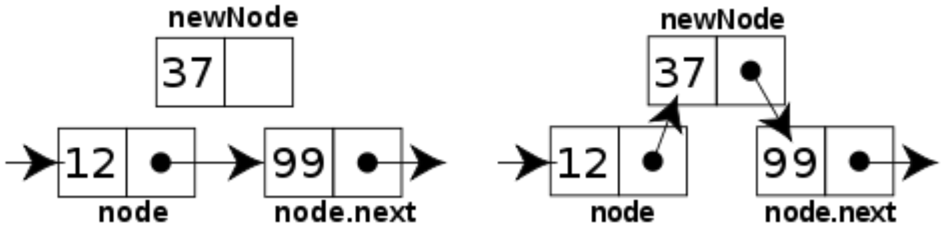
Olivia (LA)

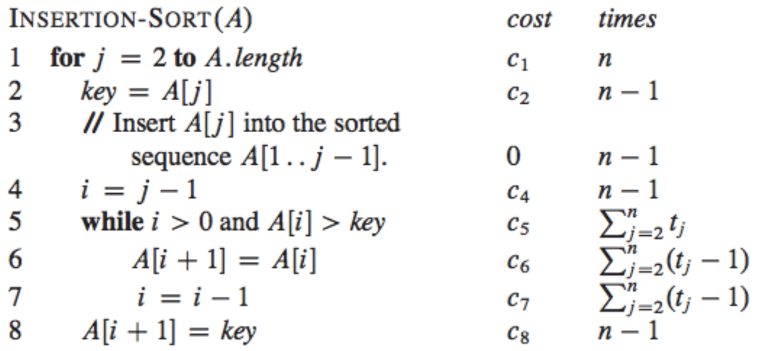
About this course:

Fast coding, clear thinking, no AI shortcuts

- Design and implement **larger programs** that **run fast**
  - Organize **data** in programs using **data structures**
  - **Analyze** the **complexity** of your programs
- Prep for **technical interviews**
- **Today: Solve a classic problem to reverse a linked list**







```
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 $A[i] > key$               $c_5$   $\sum_{j=2}^{n} t_j$
          $A[i+1] = A[i]$                          $c_6$   $\sum_{j=2}^{n} (t_j - 1)$
          $i = i - 1$                              $c_7$   $\sum_{j=2}^{n} (t_j - 1)$
      $A[i+1] = key$                               $c_8$   $n-1$
```

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

Course Logistics

- Course website: https://ucsb-cs24.github.io/s25
  - schedule, assignments, course setup
- Read the syllabus.
- Today: I'll focus on the *why* behind the course policies

LeetCode = interview practice; AI = learning tool only

Graded Components

- **Leet Code** + **Mock interview**: 10%
  - 10 medium problems from assigned problem set by week 9
  - At least one mock interview with an LA/TA by week 10
  - Why 10 LeetCode problems? They mirror interview questions—solving them builds the skills companies test.

- **Programming assignments:** 30%
  - includes shorter lab assignments + more complex programming assignments

- **Midterm:** 25% (on **05/08** during regular lecture time)

- **Final Exam:** 35% (on **06/09**, noon - 3p)
  - Final exam threshold: 65% on the final exam is required to pass the class (why threshold?)

How to succeed in this course

- **Success tip:** Own your learning— read before lectures, attend, stay on assignments (see website schedule), and ask for help in office hours.

- **AI = learning tool only:** You may use AI to understand material (e.g., ‘Explain heaps’), not to write your code—logs required when allowed. It’s about your growth, not shortcuts.

- **Why limit AI?** In interviews, you won’t have it—you’ll need to reason and code solo. Use it to learn, not to solve, and log it when allowed

- **Why integrity?** Discuss with peers, cite help, but code solo unless paired—cheating undermines your future and there will be consequences in this class for cases of plagiarism.

Preparing for lectures

- **Success tip:** Own your learning— **read before lectures**, attend, stay on assignments (see website schedule), and ask for help in office hours.

- **Prep with assigned reading before lectures**—come ready to solve problems.
  - **DS: *Data Structures and Other Objects Using C++* (Savitch, 4th ed.)**
  - **OP: *Open Data Structures* by Pat Morin (Free)**
    - https://opendatastructures.org/ods-cpp/Contents.html
  - **Dasgupta: *Algorithms* by Dasgupta & Vazirani**

About lectures

- Lectures aren’t textbook recaps—they’re problem-solving sessions. Ask questions, discuss with neighbors, answer via iClicker.

- Why interactive? You learn by doing—e.g., tracing pointers, mock interview today.

- Take a moment to introduce yourself to the people sitting near you.
    - Talk about…
        - your background,
        - experience in CS so far, and
        - what you hope to get out of this class!
        - A fun thing that you did over Spring break

About you: When did you take CS16 or an equivalent course?

A. Fall 2024

B. Summer 2024

C. Spring 2024

D. Winter 2024 or earlier

- **Why iClicker?** Join at https://join.iclicker.com/AXZR
  - its practice, not points,
  - to engage with concepts like today’s linked list

About you…

What is your familiarity/confidence in 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 git or any version control system?

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.

Remember to:

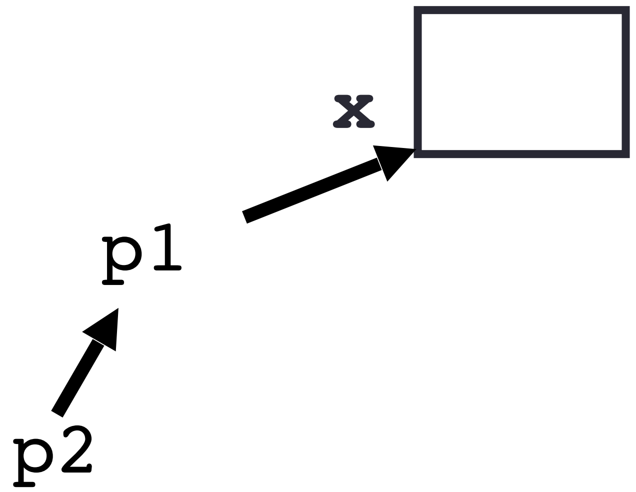
1) accept the invitation sent to your @umail.ucsb.edu account to join the class GitHub Organization (org): **ucsb-cs24-s25**
2) If unfamiliar with git complete optional lab00 by Friday

Review: Pointer assignment

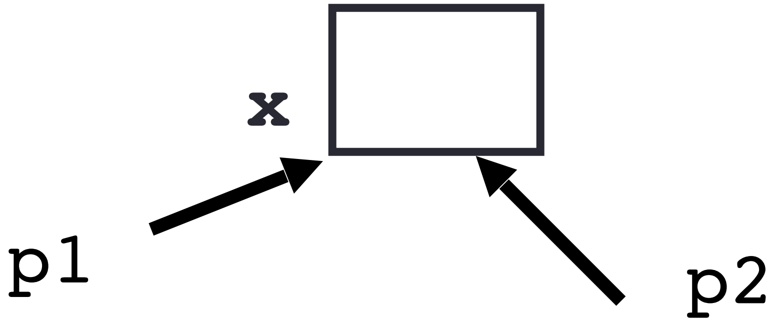
```
int* p1, *p2, x;
p1 = &x;
p2 = p1;
```

Q: Which of the following pointer diagrams best represents the outcome of the above code?

A.



B.



C. Neither, the code is incorrect

Linked list vs Array

Arrays: fixed, fast access. Linked lists: dynamic, flexible inserts.

Why care? CS24 picks the right tool — lists next!

Draw both long form and short hand representation of a linked list

Array

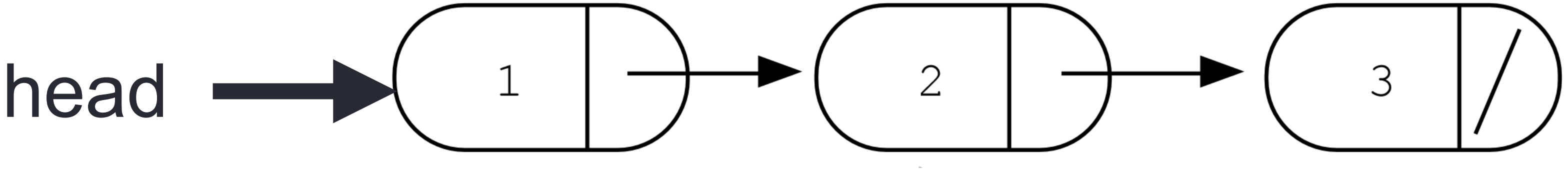
Review: Accessing structs using pointers

```
Node n {20, nullptr};
Node m {10, nullptr};
Node *p = &m;
```

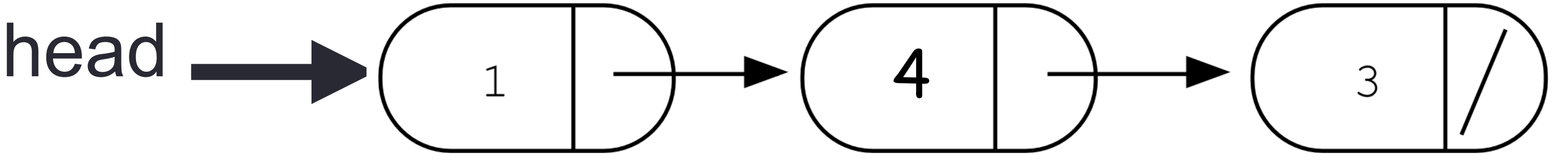
How does the given code modify the provided linked list?

```
Node* p = head;
p→next;
p→data = 4;
```

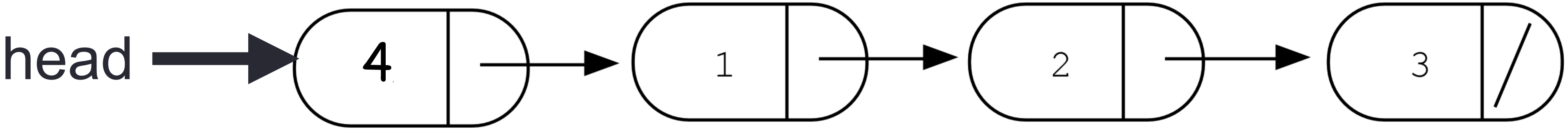
```
struct Node {
        int data;
        Node* next;
};
```



A.



B.



C. Something else

LeetCode Problem

- Reverse a linked list — classic interview task. Let’s think, explain, code it
- Problem: reverse 1→2→3 to 3→2→1.
- **Discuss (2 min): what does it take to impress your interviewer?**

Link to problem on LeetCode:
https://leetcode.com/problems/reverse-linked-list/description/?envType=problem-list-v2&envId=linked-list

Problem Clarification (2 min)

Interview Tip:

- Understand the problem and any constraints:
  - Is this singly linked? (Yes.) Any other constraints?
  - Draw the input linked list and the desired output

LeetCode Problem: Initial Exploration (3 min)

Interview Tip:

- The most important thing is to show **a logical progression of ideas**
- **Think out loud**—interviewers love hearing your reasoning.
  - Start simple, even if it’s wrong—it shows process.
- Think: How do we flip links? Try it—discuss with neighbors(3 min).

Iterative solution (7 min)

Interview Tip(s):

- Start iteratively — why?
- One iterative change — why?
- False starts are okay — why?
- Think out loud!

With one pointer curr starting at 1 in 1→2→3, what's your first step to reverse it iteratively?
A) Set curr→next to null (1→null)
B) Set curr→next→next to curr (2→1)
C) Move curr to 2, then set 1→null
D) Set curr to point to 3 (skip to end)
Identify the challenges with using just one pointer, then work towards a better solution

Iterative solution (7 min)

Discuss:

- Start iteratively — its the most natural way to traverse a pointer
- One iterative change?
- Useful pointers?
- Think out loud!

Lets's code the iterative solution

Recursive solution (7 min)

- Recursive Reverse - Same goal: 1→2→3 changes to 3→2→1
- (2 min) Discuss your solution with your neighbors
    - Discuss base case then recursive case

What's the base case for recursion?
A) Empty or one node list
B) One node list only
C) Empty list only only
D) Always recurse

Recursive case

In an interview, how should you explain the recursive step for input 1→2→3?
A) “I keep calling the function until I hit the end, then reverse everything.”
B) “I recurse on 2→3 to get 3→2, then set 2’s next to 1 and 1’s next to null.”
C) “I swap 1 and 3 in one step, leaving 2 in the middle.”
D) “I make 3 point to 2, then stop because it’s reversed.”
E) “I move the head to the end of the list, then recurse”

Let’s code the recursive solution.
In an interview we would wrap up by discussing space-time tradeoffs (next lecture: analyzing running time)

Questions for office hours

- Reflect on today’s lecture and note down at least one question that you would ask during the upcoming office hours (1 min)

Next time

- We’ll analyze the speed of programs
- Be sure to do the required reading listed on the course website