# FINAL PRACTICE!

Problem Solving with Computers-II

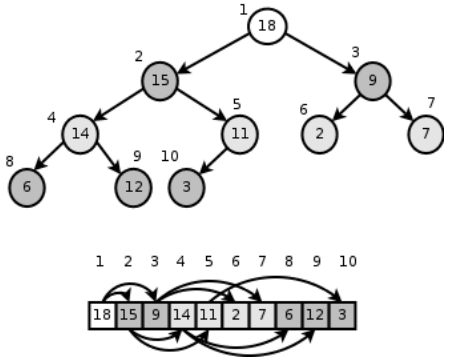

Link to class handout:

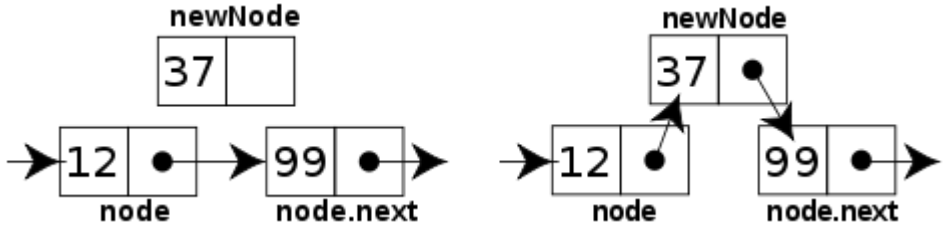
https://bit.ly/one-problem-to-rule-them-all

# I can deal with pressure, and deadlines.

# Resources for the Final Exam

- Leet code problems categorized to help you focus on the most relevant ones.
  - Those marked as “Optional challenge” won’t be on the final
- Review session led by LAs and TAs, this Sunday (over Zoom) at 10a
- Code from lectures: https://github.com/ucsb-cs24-s25/cs24-s25-lectures
- Past Exams: Available on Canvas
- Tool to visualize data structures: https://visualgo.net/







**Data Structures and C++**

| 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 $A[i+1] = key$ | $c_8$ | $n-1$ |

**Complexity Analysis**

# One problem to rule them all!

**Problem (LP04): Longest Consecutive Sequence**

**Description:** Given an unsorted array of integers `nums`, return the length of the longest consecutive elements sequence.

**Try solving it in as many different ways as you can, then pick the best!**

**Ex:** `nums = [100, 4, 200, 1, 3, 2]` `nums = [1, 0, 1, 2]`

# Approach 1

```
nums = [100, 4, 200, 1, 3, 2]
```

```
nums = [1, 0, 1, 2]
```

# Approach 2

```
nums = [100, 4, 200, 1, 3, 2]
```

```
nums = [1, 0, 1, 2]
```

# Approach 3

```
nums = [100, 4, 200, 1, 3, 2]
```

# Approach 4 ?

# Reflection + Q&A

- Which approach did you try first, and why?
  - Did exploring other methods change how you thought about the problem?

- Which tool or idea resonated most with you?
  - If you had to solve this under time pressure, which approach would you pick and why?

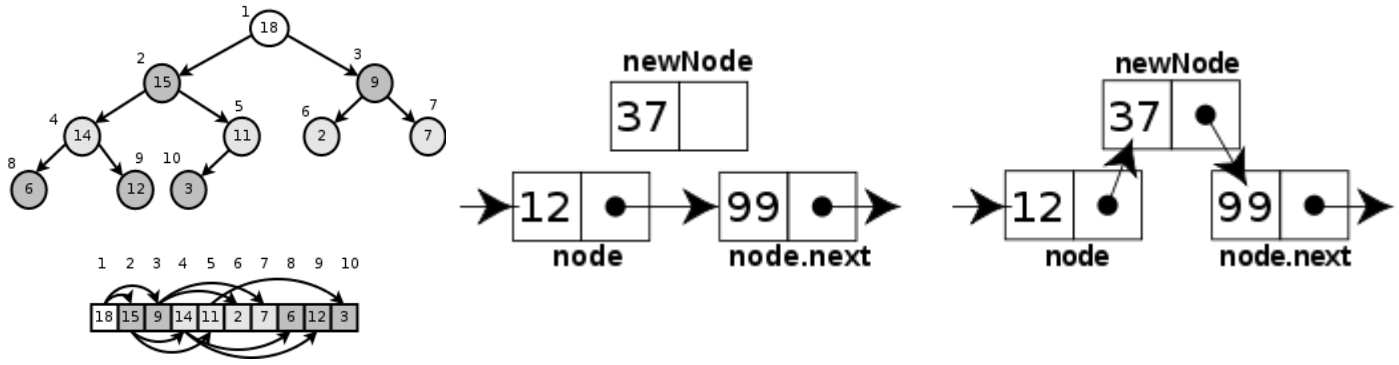
Questions?

# CS24: Learning goals

Correct 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**

**Grow confident in your problem solving skills!**




| 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**

## Break: Please take a moment to fill the course evaluations!

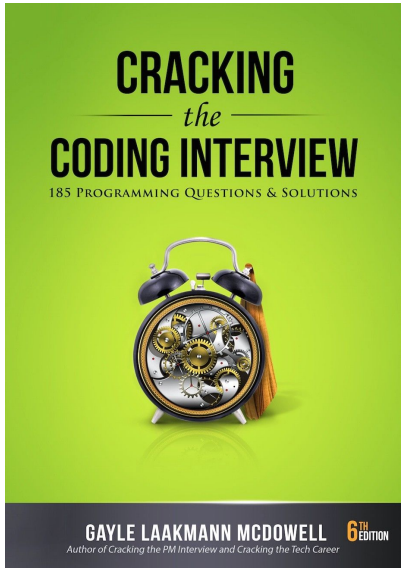
# PROBLEM SOLVING II

Student-FO

https://go.blueja.io/WD3FDfJ2QUqKjETiskpCNw

To access the evaluation, scan this QR code with your mobile phone.

# Tips for Technical Interviews




1. Listen carefully
2. Draw an example
3. State the brute force or a partially correct solution
    - then work to get at a better solution
4. Optimize:
    - Make time-space tradeoffs to optimize runtime
    - Precompute information: Reorganize the data e.g. by sorting
5. Solidify your understanding of your algo before diving into writing code.
6. Start coding!

# Thank you and all the best !