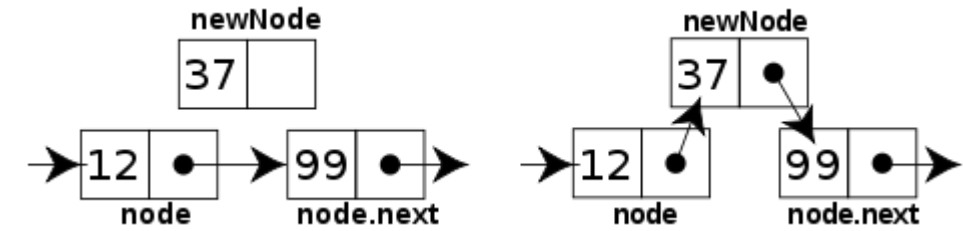


```

INSERTION-SORT(A)
1  for j = 2 to A.length
2    key = A[j]
3    // Insert A[j] into the sorted
      sequence A[1..j - 1].
4    i = j - 1
5    while i > 0 and A[i] > key
6      A[i + 1] = A[i]
7      i = i - 1
8    A[i + 1] = key

```

cost	times
$c_1$	$n$
$c_2$	$n - 1$
$c_3$	$n - 1$
$c_4$	$n - 1$
$c_5$	$\sum_{j=2}^n t_j$
$c_6$	$\sum_{j=2}^n (t_j - 1)$
$c_7$	$\sum_{j=2}^n (t_j - 1)$
$c_8$	$n - 1$



# WELCOME TO CS 24!

## Problem Solving with Computers-II

Instructor: Diba Mirza

C++

```

#include <iostream>
using namespace std;

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

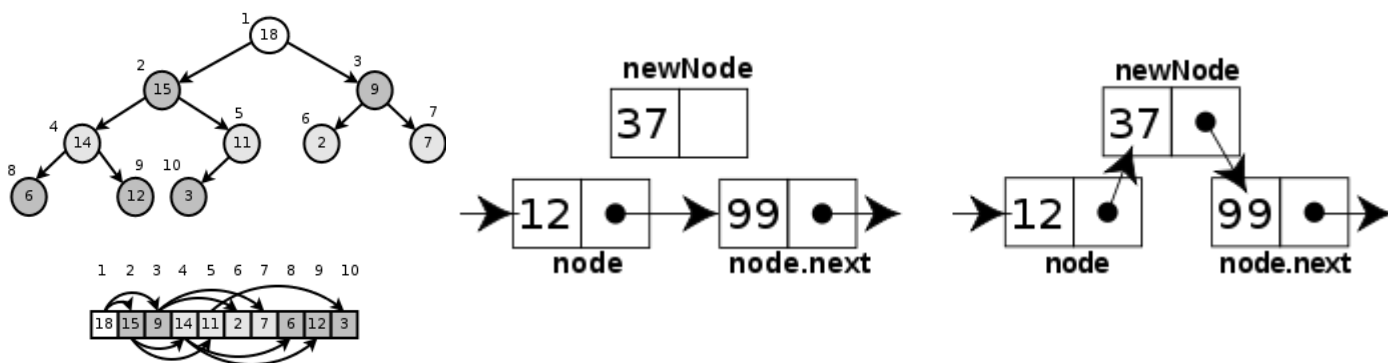
```

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)
1  for j = 2 to A.length
2    key = A[j]
3    // Insert A[j] into the sorted
   sequence A[1..j-1].
4    i = j - 1
5    while i > 0 and A[i] > key
6      A[i + 1] = A[i]
7      i = i - 1
8    A[i + 1] = key
  
```

cost	times
$c_1$	$n$
$c_2$	$n - 1$
0	$n - 1$
$c_4$	$n - 1$
$c_5$	$\sum_{j=2}^n t_j$
$c_6$	$\sum_{j=2}^n (t_j - 1)$
$c_7$	$\sum_{j=2}^n (t_j - 1)$
$c_8$	$n - 1$

## Data Structures and C++

## Complexity Analysis

# About the team



Instructor: Diba Mirza

- **TAs:** Lucas, Ganesh, Roman
- **ULAs:** Tina and Zack

- Communication with staff via **Piazza**
- Include [CS24] in the subject line of any email communication with me
- Sections start this week, office hours start next week

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

# Course Logistics

- Course 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](mailto: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



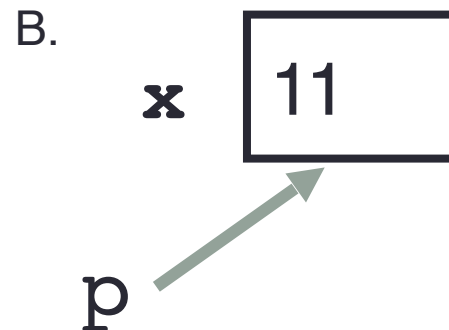
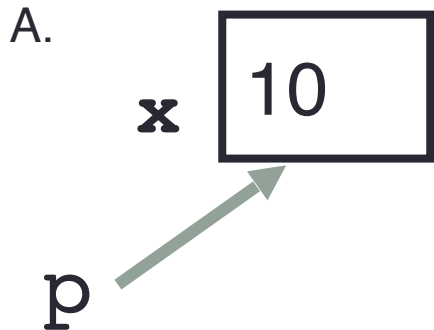
# About lectures

- 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!

# Review: Tracing code involving pointers

```
int* p;  
int x = 10;  
p = &x;  
*p = *p + 1;
```

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

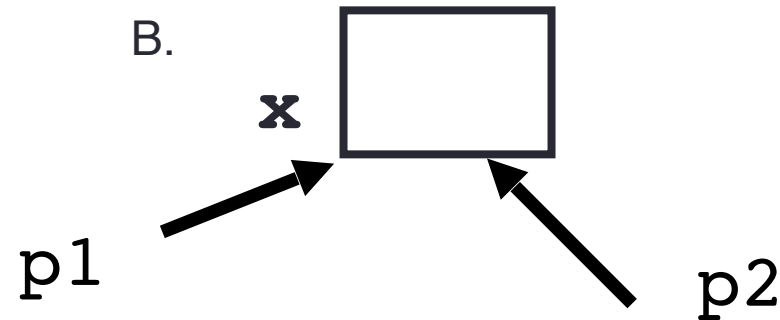
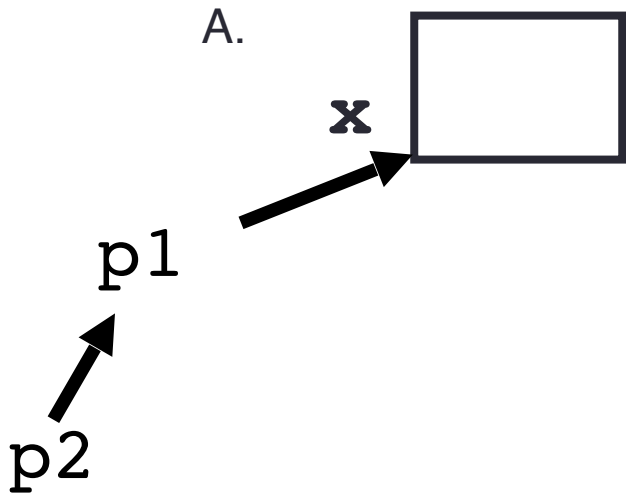


C. Neither, the code is incorrect

# 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?



C. Neither, the code is incorrect

# Two important facts about Pointers

1) A pointer can only point to one type –(basic or derived ) such as `int`, `char`, a `struct`, a `class` another pointer, etc

2) After declaring a pointer: `int *ptr;`  
`ptr` doesn't actually point to anything yet.

We can either:

- make it point to something that already exists, OR
- allocate room in memory for something new that it will point to

# Review: Heap vs. stack

```
1 #include <iostream>
2 using namespace std;
3
4 int* createIntArray(int len){
5
6     int arr[len];
7     return arr;
8
9 }
```

Where does the above function create the array of integers?

A. Stack

B. Heap

C. Don't know, what do you mean by stack and heap?

# Review: C++ Program's Memory Regions

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

// Program is stored in code memory

int myGlobal = 33;    // In static memory

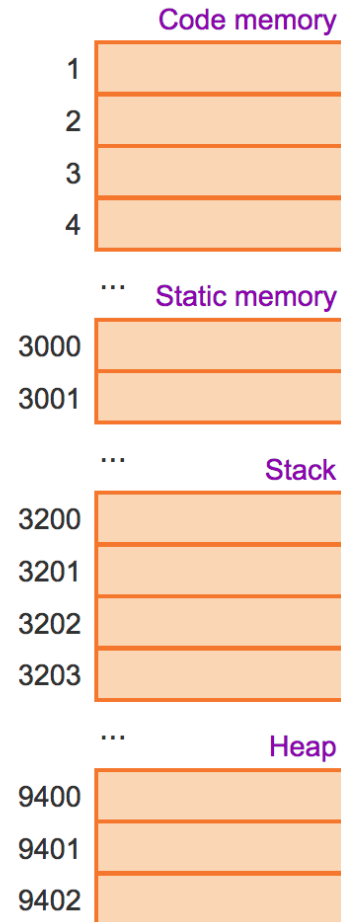
void MyFct() {
    int myLocal;      // On stack
    myLocal = 999;
    cout << " " << myLocal;
}

int main() {
    int myInt;         // On stack
    int* myPtr = nullptr; // On stack
    myInt = 555;

    myPtr = new int;    // In heap
    *myPtr = 222;
    cout << *myPtr << " " << myInt;
    delete myPtr; // Deallocated from heap

    MyFct(); // Stack grows, then shrinks

    return 0;
}
```



The code regions store program instructions. `myGlobal` is a global variable and is stored in the static memory region. Code and static regions last for the entire program execution.

# Review: C++ Program's Memory Regions

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

// Program is stored in code memory

int myGlobal = 33;    // In static memory

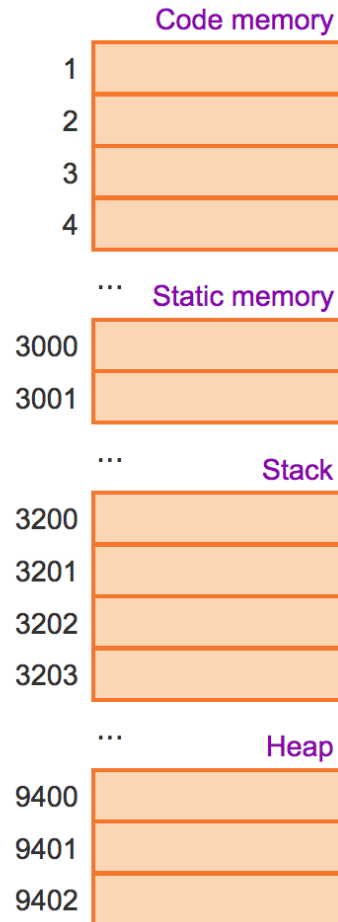
void MyFct() {
    int myLocal;      // On stack
    myLocal = 999;
    cout << " " << myLocal;
}

int main() {
    int myInt;         // On stack
    int* myPtr = nullptr; // On stack
    myInt = 555;

    myPtr = new int;    // In heap
    *myPtr = 222;
    cout << *myPtr << " " << myInt;
    delete myPtr; // Deallocated from heap

    MyFct(); // Stack grows, then shrinks

    return 0;
}
```



- Stack: Segment of memory managed automatically using a Last in First Out (LIFO) principle.
- Heap: Segment of memory managed by the programmer
  - Data created on the heap stays there
  - FOREVER or
  - until the programmer explicitly deletes it

The code regions store program instructions. `myGlobal` is a global variable and is stored in the static memory region. Code and static regions last for the entire program execution.

# Next time

- Linked lists.