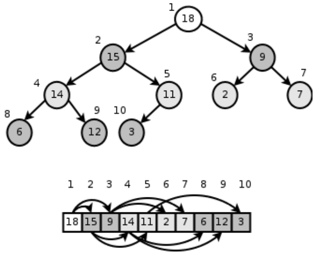
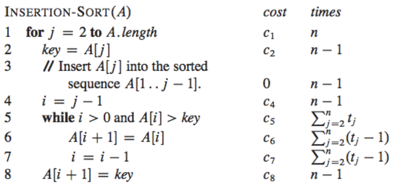
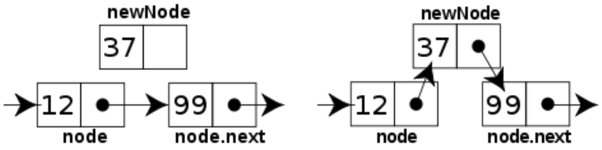




```
INSERTION-SORT(A)                               cost    times
for j = 2 to A.length                           $c_1$   $n$
    key = A[j]                                  $c_2$   $n-1$
    // Insert A[j] into the sorted
            sequence A[1 . . j − 1].            0       $n-1$
    i = j − 1                                   $c_4$   $n-1$
    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;
}
```

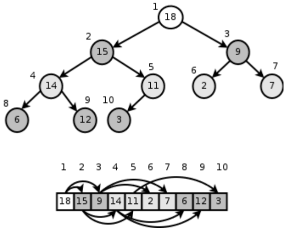
Read the syllabus. Know what’s required. Know how to get help.

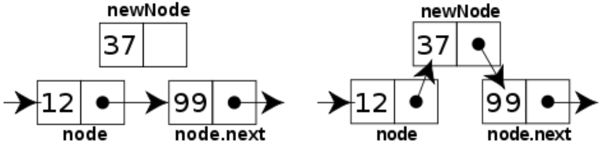
About this course

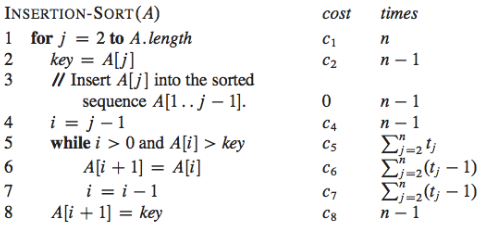
C++ fast

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**







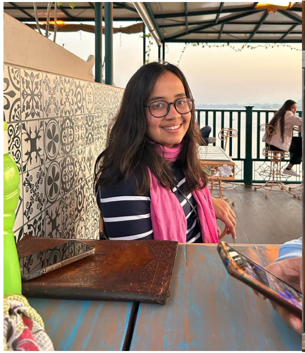
```
INSERTION-SORT(A)                                   cost   times
for j = 2 to A.length                               $c_1$    $n$
    key = A[j]                                      $c_2$    $n-1$
    // Insert A[j] into the sorted
         sequence A[1..j-1].                        0      $n-1$
    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

About the team

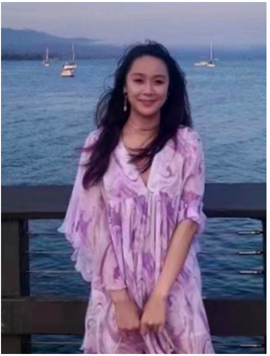
Diba Mirza



TAs: Samridhi

Kaiwen

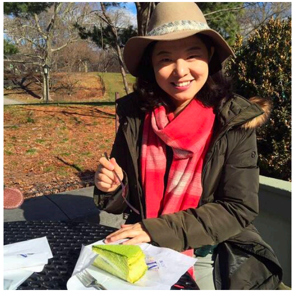
Tyler



Evelyn

- Prof. Mirza’s OH:
  MW 2:30-3:30p, HFH 1155
- 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.



TA: Lijuan

ULAs: Zack

Rachel

Course Logistics

- Course website: https://ucsb-cs24.github.io/w23
- If you have a section conflict, you may informally switch your section time.
- NO MAKEUP ON EXAMS!
- Submit assignments early to 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

1. Login: https://app.reef-education.com/#/login
2. Join the class: CMPSC24: Problem Solving with Computers-2

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: When did you take CS16?

A. Fall 2022

B. Summer 2022

C. Spring 2022

D. Sometime before Spring 2022

About you: How was your experience in CS16?

A. Great! I enjoyed the course.

B. A little rocky. I struggled a bit but was able to get help when needed.

C. I struggled a lot but felt connected to the staff and my peers

D. I struggled a lot

How confident do you feel about CS16 topics?

A. Very confident

B. Somewhat confident

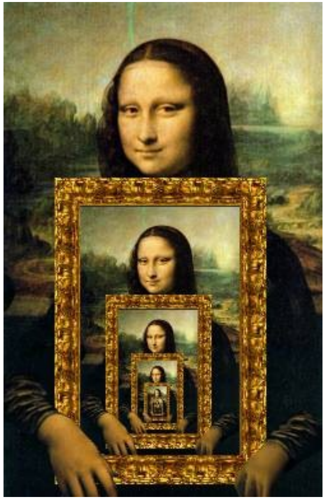
C. Not confident

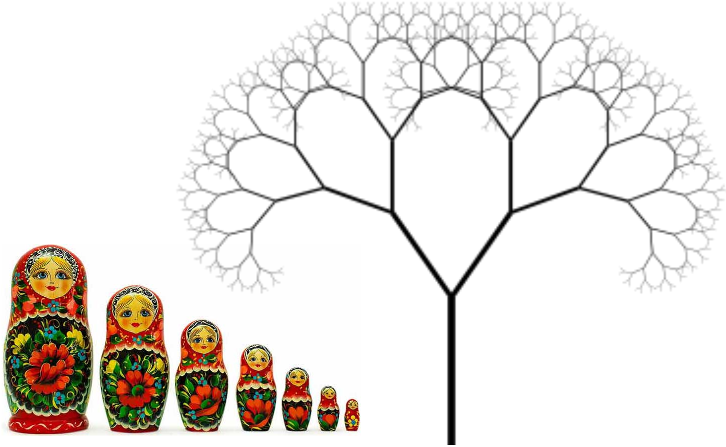
About lectures

- I will not be a talking textbook
- 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,
        - experience in CS so far, and
        - what you hope to get out of this class!

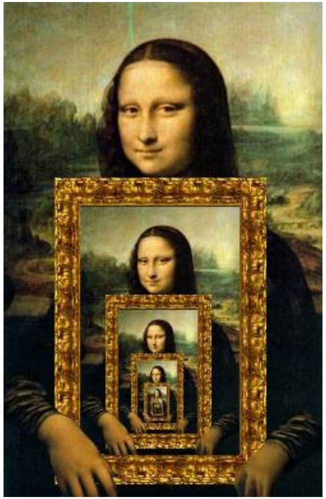
Review: Recursion

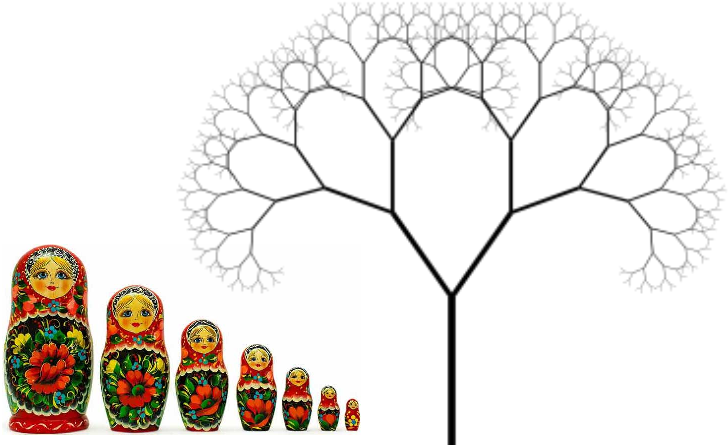




Review: Recursion

- Solve the simplest case of the problem
- Solve the general case by describing the problem in terms of a smaller version of itself





Factorial

$$3! = 3 * 2 * 1$$

$$\boxed{4!} = 4 * \underbrace{3 * 2 * 1}$$

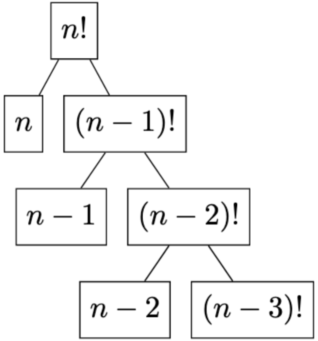
$$= 4 * \underline{3!}$$

$$\underline{n}! = n * \underline{(n-1)}!$$

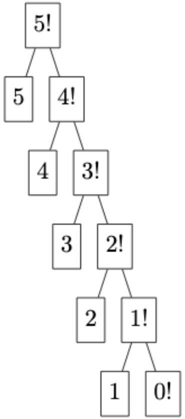
Thinking ***recursively***

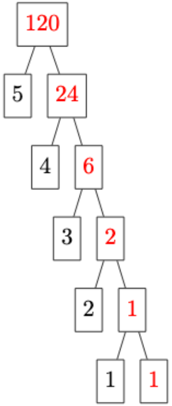
```
N!  =  N * (N-1)! , if  N > 1
    =  1, if N <= 1
```

Recursion == ***self***-reference!



Computing a recursive function





Designing Recursive Functions

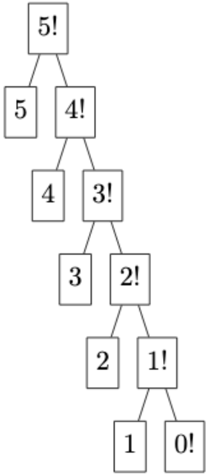
```
int fac(int N){
    if(N <= 1){
        return 1;
    }




}
```

Base case:

Solution to inputs where the answer is simple to solve



Designing Recursive Functions

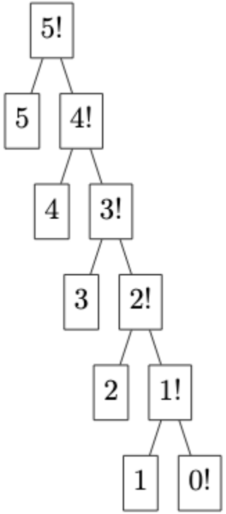
```
int fac(int N){
    if(N <= 1){
        return 1;
    }

    return N* fac(N-1);


}
```

Base case

Recursive case



Human: Base case and 1 step

Computer: Everything else

Warning: *this is legal!*

```
int fac(int N){
    return N* fac(N-1);
}
```

legal != *recommended*

```
int fac(int N){
    return N* fac(N-1);
}
```

No ***base case*** -- the calls to `fac` will never stop!

Make sure you have a **base case**, *then* worry about the recursion...

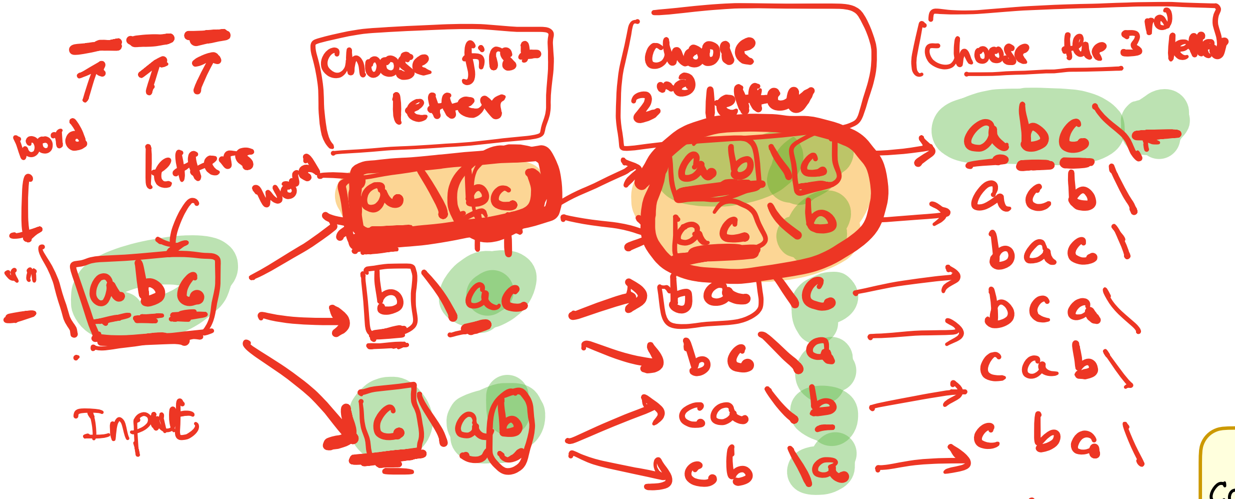
Word Scrambling

permutation

Recursively print all scrambling of the letters

“abc”

Base case



not returning a single value

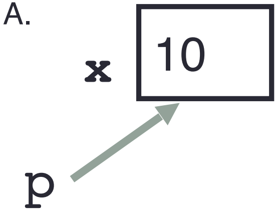
Coding demo

Review: Tracing code involving pointers

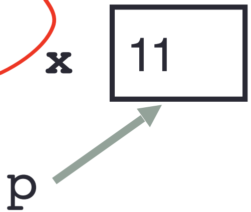
```
int* p;
int x = 10;
p = &x;
*p = *p + 1;
```

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

A.



[✓ B.]



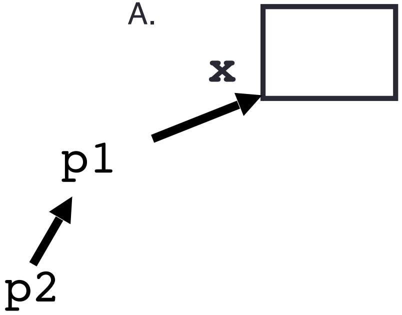
C. Neither, the code is incorrect

Review: Pointer assignment

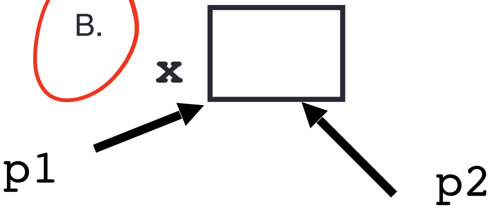
```
int* p1, *p2, x;
p1 = &x;
p2 = p1;
```

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

A.



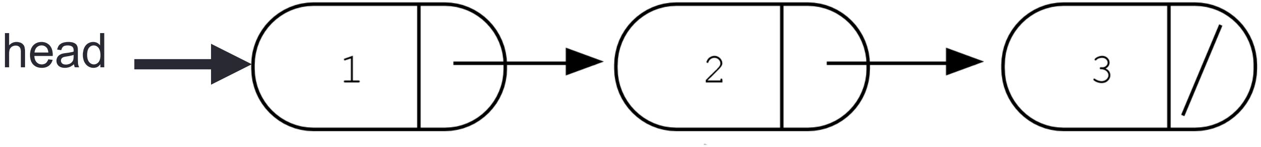
[✓ B.]



C. Neither, the code is incorrect

Assume the following linked list exists

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

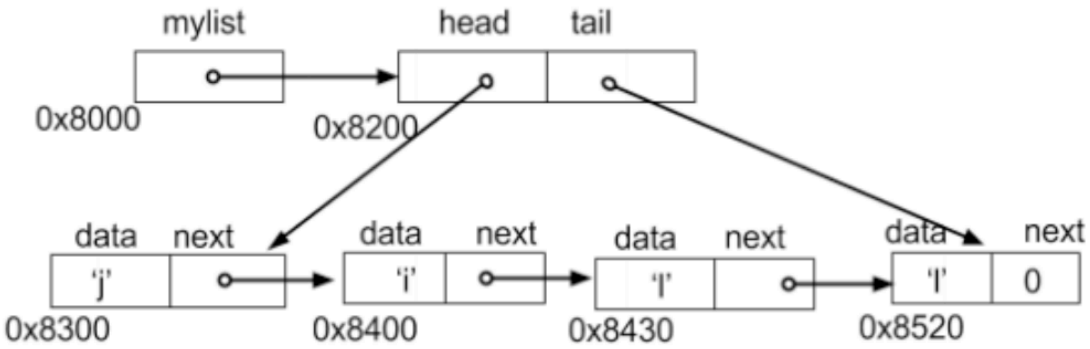


Evaluate each of the following expressions?

```
1. head->data
2. head->next->data
3. head->next->next->data
4. head->next->next->next->data
```

A. 1
B. 2
C. 3
D. nullptr
E. Run time error

Accessing nodes in a linked list



a. cout<<mylist;

b. cout<<mylist->tail;

c. cout<<mylist->tail->data;

d. cout<<mylist->head->next;

e. cout<<mylist->head->next->

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

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

int* createAnIntArray(int len){

    int arr[len];
    return arr;

}
```

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?

Next time

- We’ll solve the final exam for CS16 (Fall 2022)
- Bring your laptops to class!