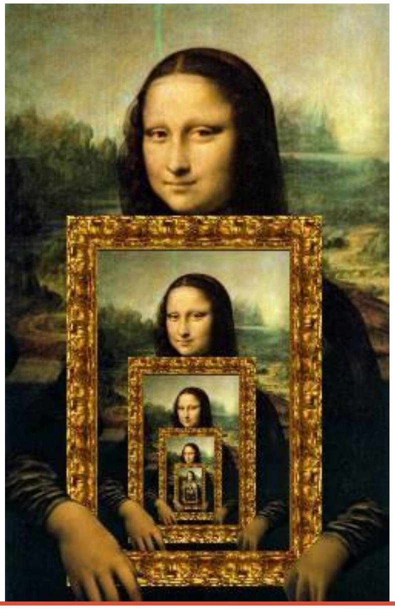
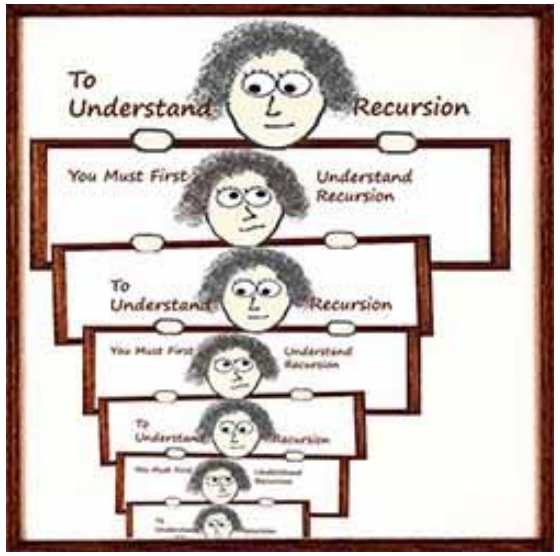
RECURSION AND FINAL REVIEW



Problem Solving with Computers-I

https://ucsb-cs16-wi17.github.io/



Final Exam!

- Thursday (03/23) 4pm to 7pm NH 1006 and CHEM 1171
- Assigned seating will be posted on Piazza
- Everything we have covered so far is on the exam
- Duration: **3 hours**
- Closed book: no calculators, no phones, no computers
- Only 1 sheet (***double***-sided is ok) of written notes
  - Must be no bigger than 8.5” x 11”
  - You have to turn it in with the exam

How do you feel after the last two labs ?

A

B

C

D

E

Lab 08: When should you use a helper function?

C string

```
bool isPalindrome(const char *s1) //recursive
    deTartraTED '\0'
    WasItACarOrACatISaw '\0'
```

This is not a string because it is not null terminated, so our recursive function should take a char array & its length as input — use helper

If the first & last characters are equal, recurse on second to second last characters

Do we need a helper function for a recursive implementation of the above function?

[✓ A. Yes]
B. No

Because we have identified that we need to pass a different set of parameters (char array, len of array) to the recursive function.

Steps towards a recursive solution:

1. Identify the recursive structure in your input and or problem
2. Write the recursive step in plain English
3. Do you need a helper function?

Lab 08: Thinking recursively!

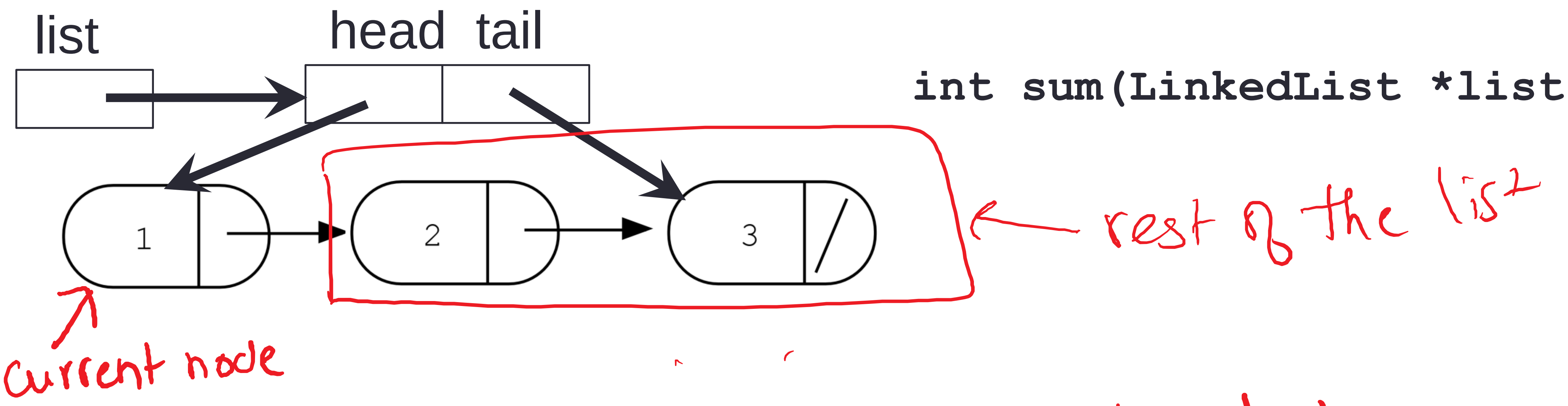
deTartraTED
WasItACarOrACatISaw

```
bool isPalindromeHelper(const char *s1, int len){
  if(len<=1)
      return true;
 else if(s1[0]==s1[len-1])
     return isPalindromeHelper(
              s1+1, len-2);
  else
     return false;
}
```

Steps towards a recursive solution:

1. Identify the recursive structure in your input and or problem
2. Write the recursive step in plain English
3. Do you need a helper function?
4. Implement (and test) the base case
5. Believe that you already have a correct implementation of the function that works on all smaller size inputs
6. Implement (and test) the recursive case

Recursion on lists: compute the sum of all elements



```
int sum(LinkedList *list)
```

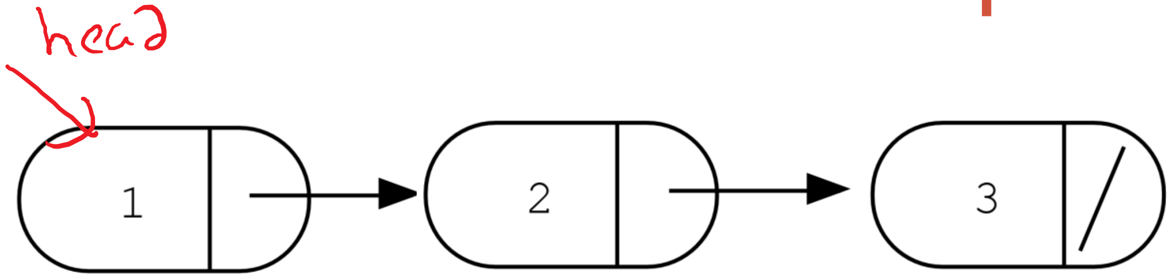
Sum of elements of current list = value of current node + sum of elements of the rest of the list

* A generic representation of "current list" is a pointer to the first node in the list (type Node *)

Use a helper function

Recall the steps towards a recursive solution

Recursion on lists: compute the sum of all elements

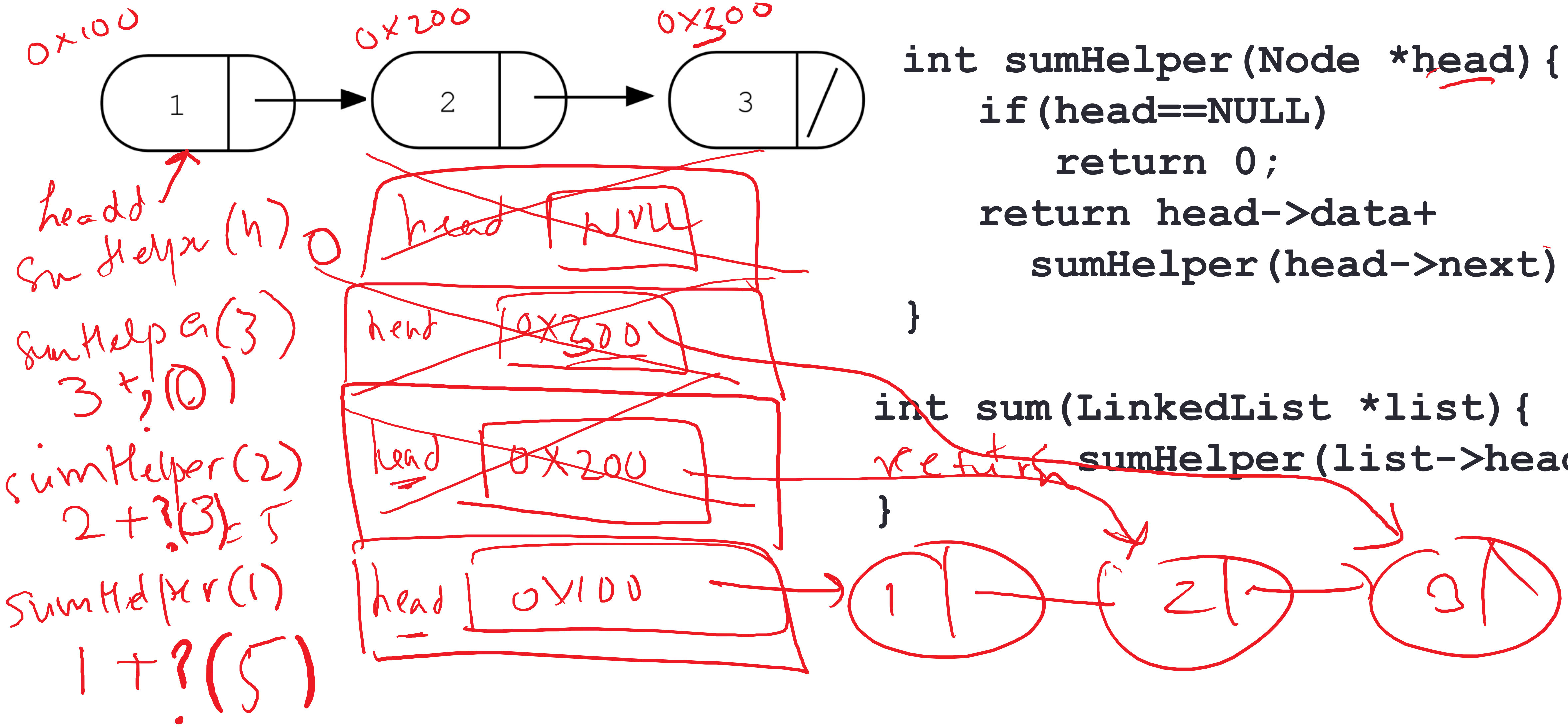


```
int sumHelper(Node *head) {
    if (head == NULL)
        return 0;
    else
        return head->data +
            sumHelper(head->next);
}
```

We believe our implementation works perfectly on all smaller size lists.

1. Identify the recursive structure in your input and or problem
2. Write the recursive step in plain English
3. Do you need a helper function?
4. Implement (and test) the base case
5. **Believe that you already have a correct implementation of the function that works on all smaller size inputs**
6. Implement (and test) the recursive case

Under the hood of recursive calls (review)

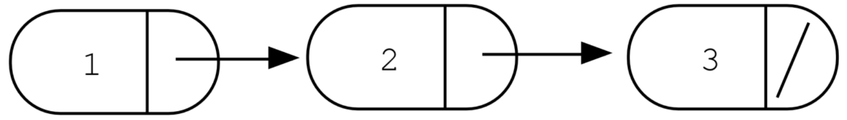


```
int sumHelper(Node *head){
    if(head==NULL)
       return 0;
    return head->data+
       sumHelper(head->next);
 }

int sum(LinkedList *list){
  return sumHelper(list->head);
}
```

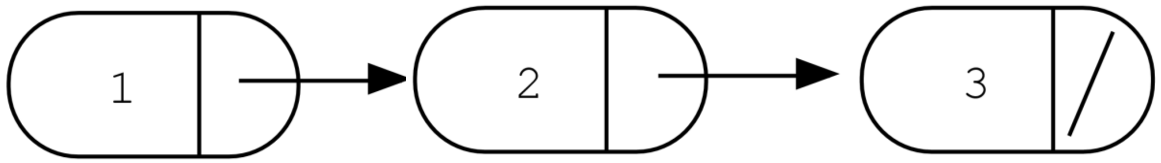
Recall the steps towards a recursive solution

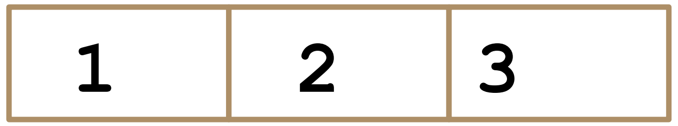
Recursion on lists: delete a value recursively



```
Node* deleteNodeRecursiveHelper(Node *head, int value)
 if(head->data == value){
 1. delete the current head node
 2. Obtain the new head for the rest of the list (by making a recursive call to
 deleteNodeRecursively() which also deletes all occurrences of value in the rest of the list)
 3. return the new head obtained from step 2
 } else{
   1. Do not delete the current node
   2. Obtain the new head for the rest of the list (by making a recursive call to
      deleteNodeRecursively(head->next) which also deletes all occurrences of value in
      the rest of the list)
  3. Make the next pointer of the current head point to the head of the rest of the list(from 2)
  4. Return the current head
}
```

How do you decide which data structure to use?





Searching for a value in a sorted array

$\leftarrow$ mid of the list

a

10	20	30	40	50	60	70	80
0	1	2	3	4	5	6	7

if value == a[mid]
found node! return

else if value you are looking for is < value of mid node
recursively search on left half of the array

else
recursively search on right half of the array

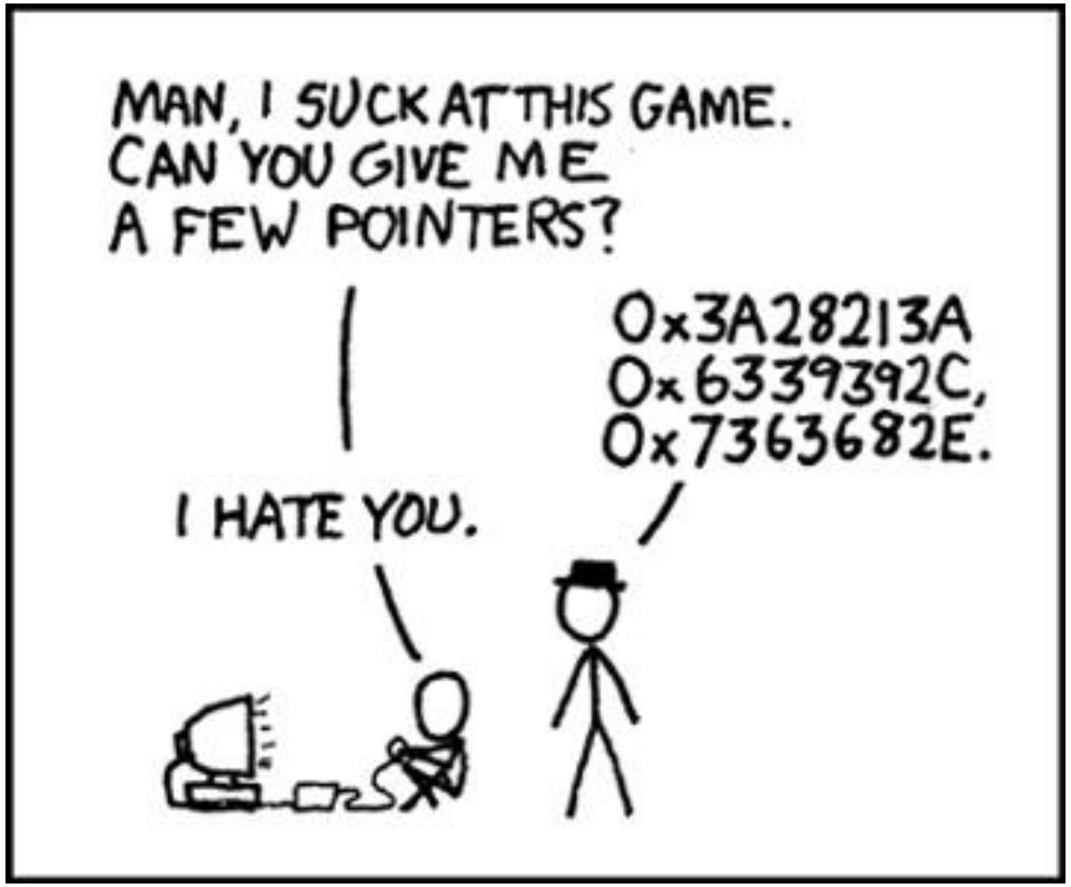
Some comic relief…

COMMENT
DATE
CREATED MAIN LOOP & TIMING CONTROL
14 HOURS AGO
ENABLED CONFIG FILE PARSING
9 HOURS AGO
MISC BUGFIXES
5 HOURS AGO
CODE ADDITIONS/EDITS
4 HOURS AGO
MORE CODE
4 HOURS AGO
HERE HAVE CODE
4 HOURS AGO
AAAAAAAA
3 HOURS AGO
ADKFJSLKDFJSDKLFJ
3 HOURS AGO
MY HANDS ARE TYPING WORDS
2 HOURS AGO
HAAAAAAAANDS
2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

HTTP://XKCD.COM/1296/

Some comic relief



HTTP://XKCD.COM/138/

Final words…

Which concepts do you need more help with?

Stay posted on Piazza

- Tutor lab hours tomorrow in CSIL
- Extended office hours next week
- Seating arrangement for exams (you may be in a different exam hall)

Final words

- You can debug your code!
- …….But you have to write it systematically!

Good luck with the final and PA8!