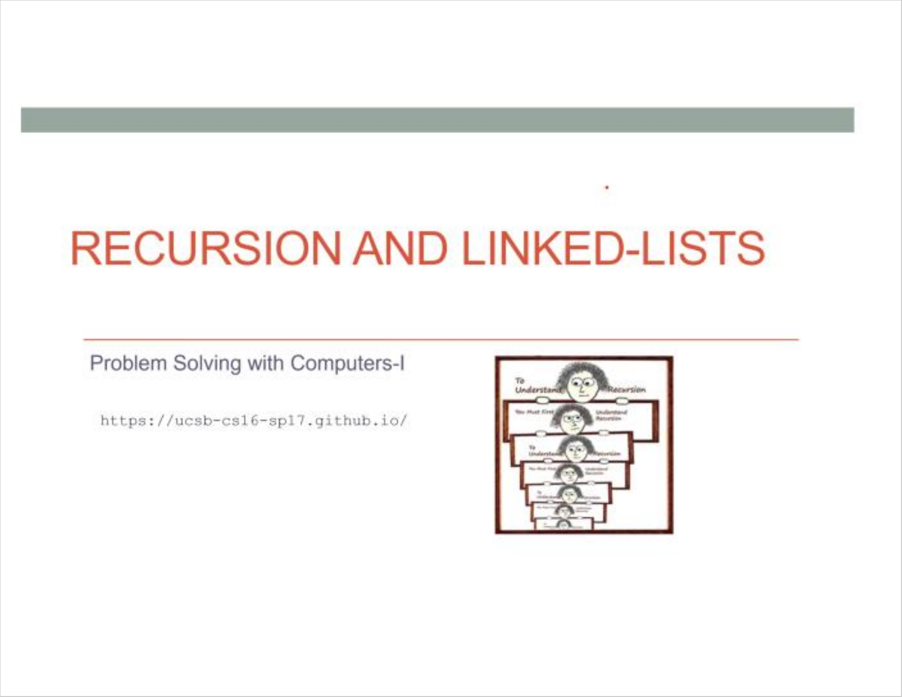
Lecture 17: More recursion with linked lists

Monday, June 5, 2017 1:42 PM



RECURSION AND LINKED-LISTS

Problem Solving with Computers-I

https://ucsb-cs16-sp17.github.io/

1. Pointers
2. Recursion
3. Linked lists

How comfortable do you feel with ___?

A. Very comfortable

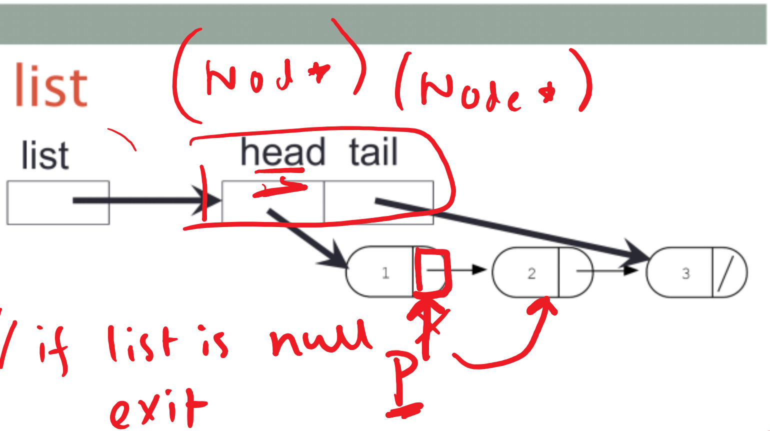
B. Have some doubts

C. Not so sure

D. Don't understand it.

Iterating through the list

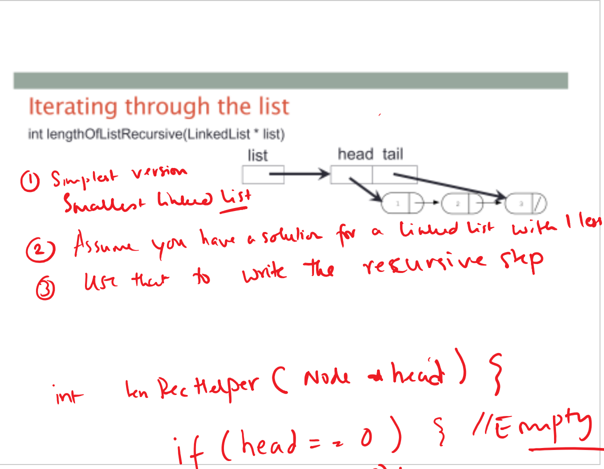
int lengthOfList(LinkedList * list)



```
int count = 0;
assert ( list ); // if list is null
                    exit
for ( Node *p = list->head; p != 0 ; p=p->next) {
        count ++ ;
   }
   return count;
}
```

Page 3

Monday, June 5, 2017 1:42 PM



Iterating through the list

int lengthOfListRecursive(LinkedList * list)

① Simplest version
Smallest Linked list

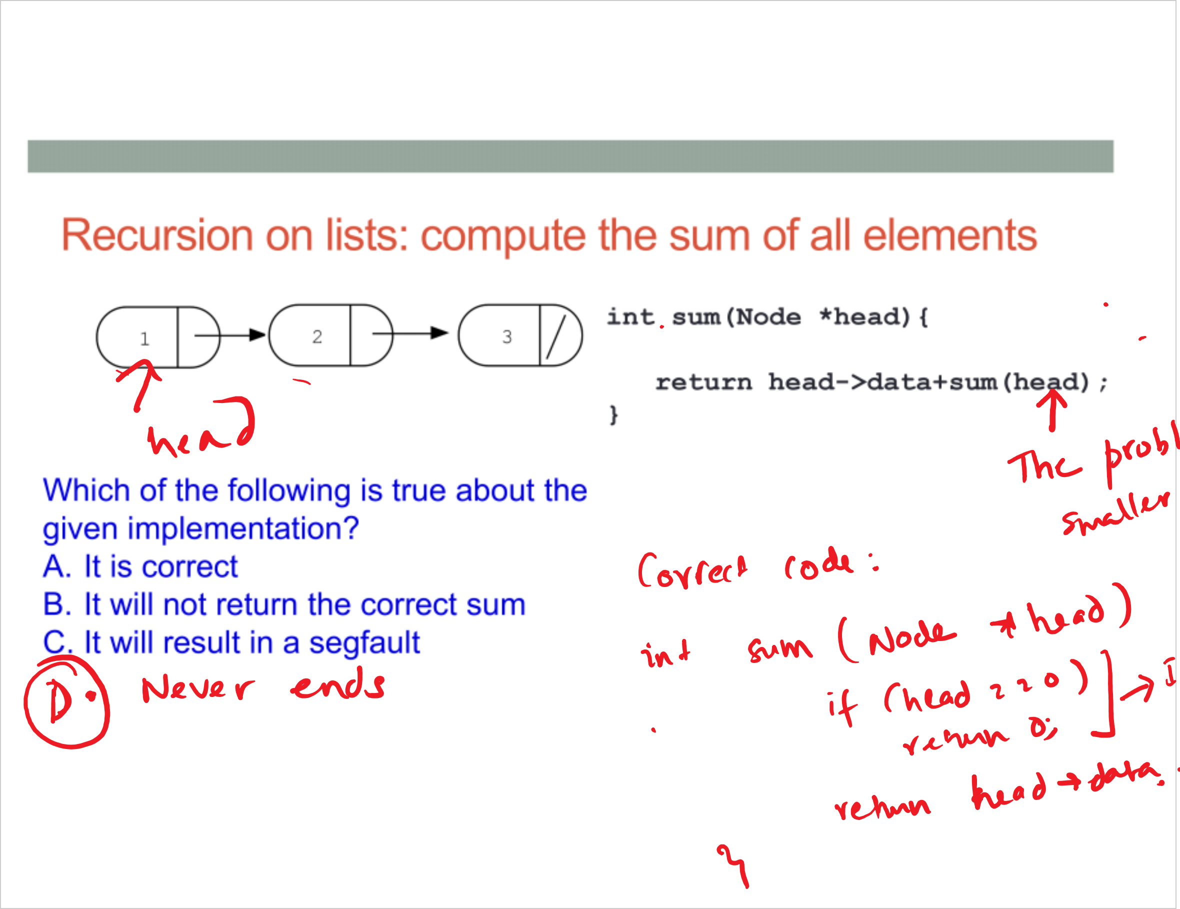
② Assume you have a solution for a linked list with 1 less node

③ Use that to write the recursive step

```
int lenRecHelper ( Node *head ) {
    if (head == 0) { //Empty list
        return 0;
    }

  int result = lenRecHelper (head->next);
     return result +1;
}
```

↑ rest of the list



Recursion on lists: compute the sum of all elements

```
int sum(Node *head){

    return head->data+sum(head);
}
```

The problem is not getting smaller

Which of the following is true about the given implementation?

A. It is correct

B. It will not return the correct sum

C. It will result in a segfault

[✓ D.] Never ends

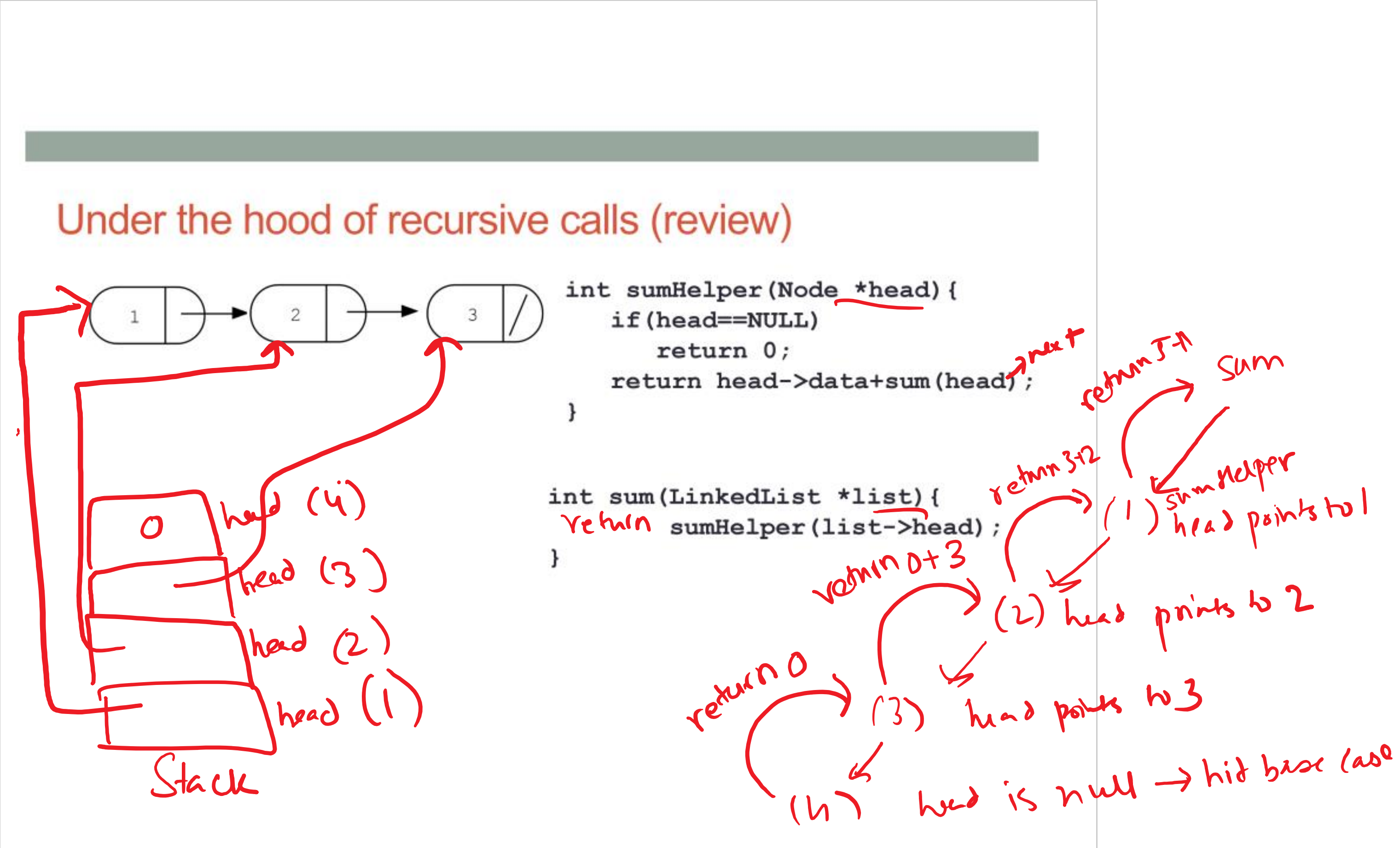
Correct code:

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

→ If no base case code will seg fault

Page 5

Monday, June 5, 2017 1:42 PM

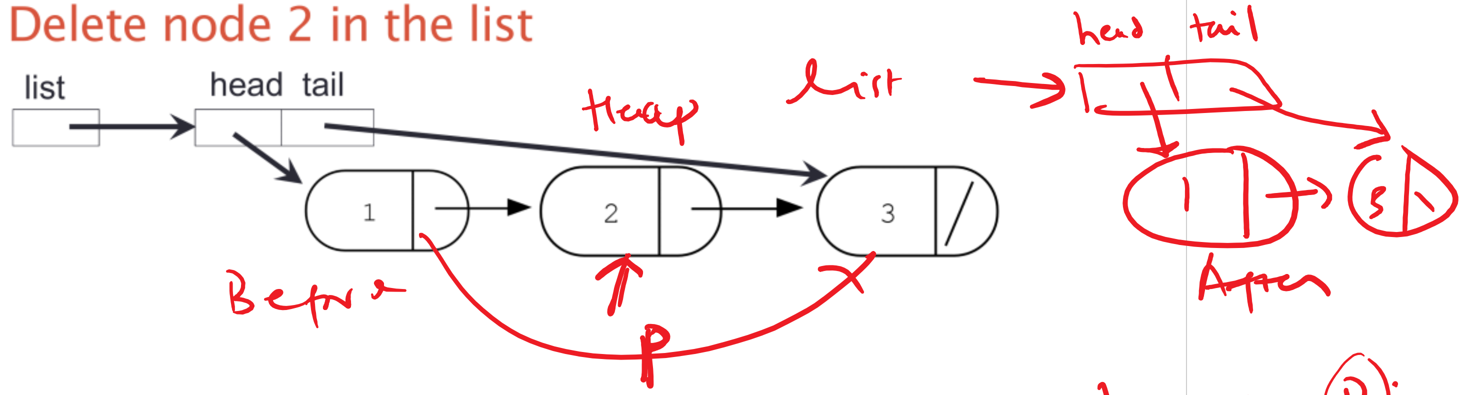


Under the hood of recursive calls (review)

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

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

Delete node 2 in the list



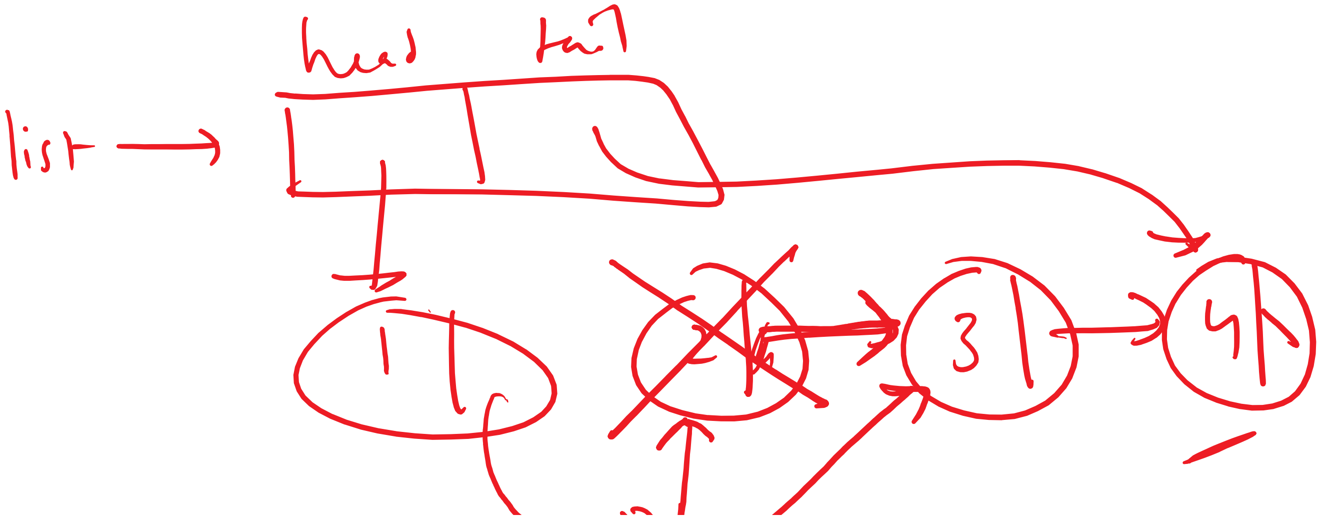
① Deallocate node 2 from heap | delete (p);

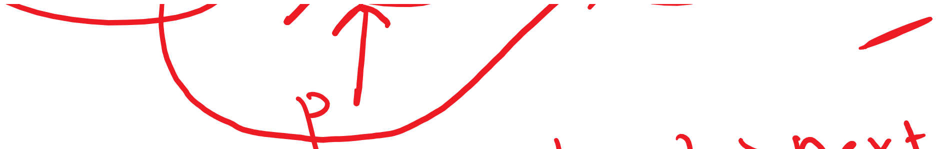
② Make the next pointer of ① point to ③

list → head → next = list → tail ;

delete list → head → next

list → head → next = list → tail;





```
Node *p = list->head->next;

        ;

list->head->next = p->next;
delete p;
```