Lecture 13: Linked Lists continued

Monday, May 22, 2017 10:13 AM

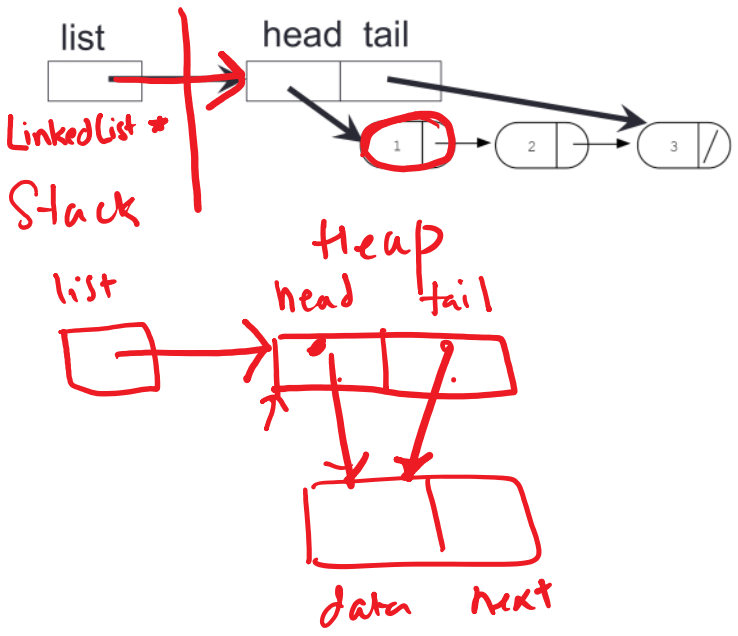
LINKED LISTS (CONTD)
DYNAMIC MEMORY PROBLEMS

Problem Solving with Computers-I

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

Monday, May 22, 2017 10:13 AM

Review:
What are the 'links' in a linked-list?



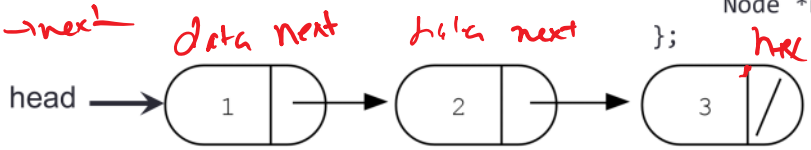

```
struct LinkedList {
    Node *head;
    Node *tail
};
struct Node {
    int data;
    Node *next;
};
LinkedList *list;
list = new LinkedList;

list->head = new Node
list->tail = list->head;
```

Monday, May 22, 2017 10:13 AM

Accessing elements of a list

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

head → data
head → next



Assume the linked list has already been created, what do the following expressions evaluate to?

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. NULL
E. Run time error → segfault

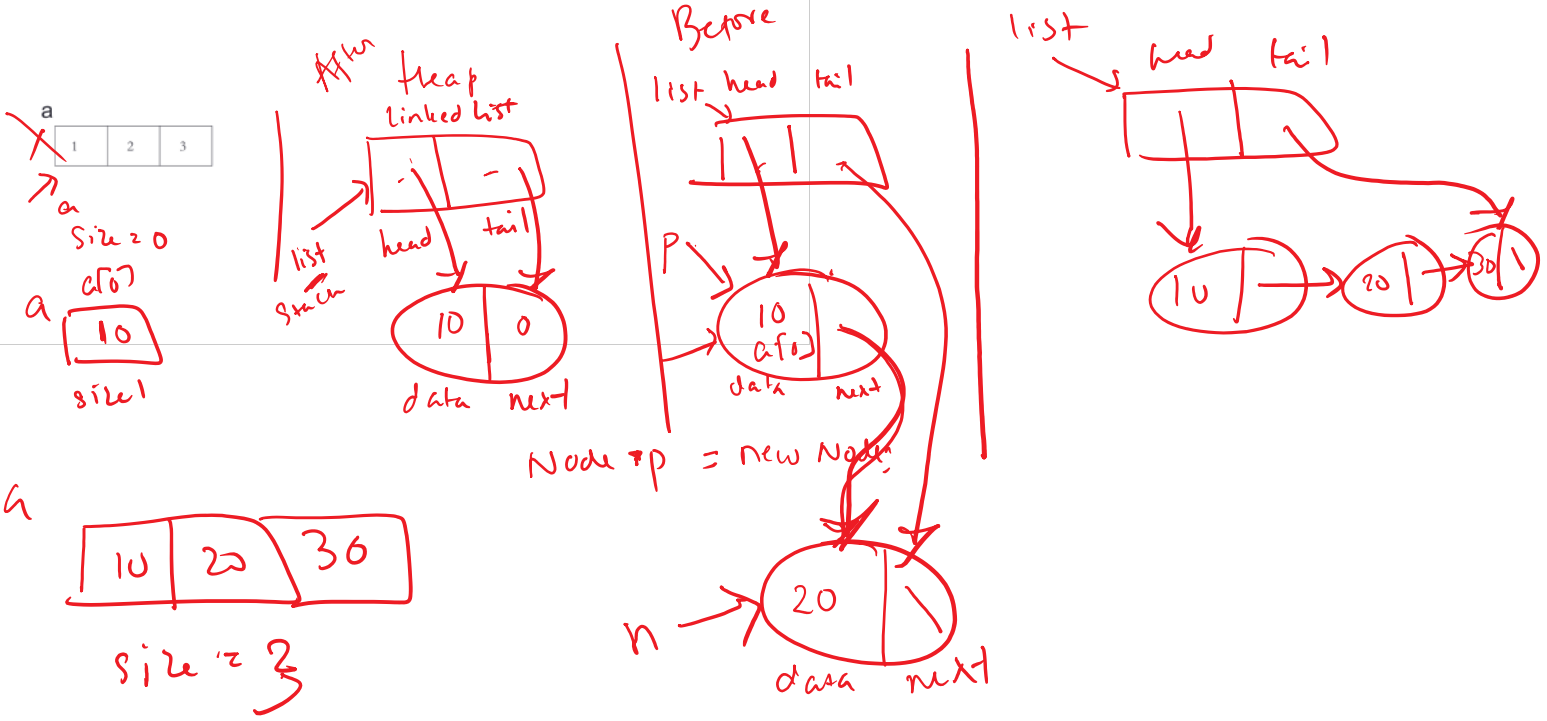
0 (null)

0 → data will result in a segfault.

In general, dereferencing a null pointer results in a segfault.

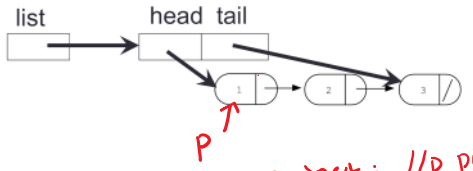
Building a list from an array

LinkedList * arrayToLinkedList(int a[], int size) ;



Iterating through the list

```
int lengthOfList(LinkedList * list) {
// Method 1: iterating through the linked list using a while loop
  int count = 0; // Initialize a counter to 0
  Node *p = list->head; // Initialize a traversal pointer p to point to the first node in the list
  while (p) {
      count++;
      p = p->next; // Move to the next node
  }
  return count;
}
```



p

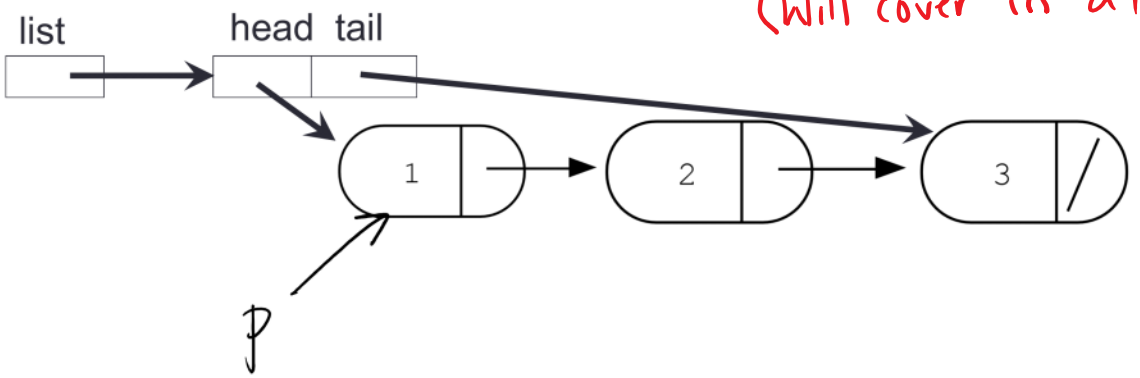
p = p->next; // p points to the next node

```
// Method 2: using a for loop
int count = 0;
for(Node *p = list->head; p!=0; p = p->next){
    count++;
}
return count;
```

Monday, May 22, 2017 10:13 AM

Delete node 2 in the list

(Will cover in a later lecture)



Delete the list

int freeLinkedList(LinkedList * list);