Week 7

Wednesday, March 17, 2021 02:53

Recitation#7: C struct and pointer: Linked List

CS232 Spring 2021

When: March 12 at 2:00 pm

Linked List is the best exercise to learn pointers and structu C

Here's one to help you in your interviews. Please go over the recitation slides first.
In `ll_cycle.c`, complete the function `ll_has_cycle()` to implement the following algorith checking if a singly-linked list has a cycle.

1. Start with two pointers at the head of the list. We'll call the first one tortoise and t second one hare.
2. Advance hare by two nodes. If this is not possible because of a null pointer, we ha the end of the list, and therefore the list is acyclic.
3. Advance tortoise by one node. (A null pointer check is unnecessary. Why?)
4. If tortoise and hare point to the same node, the list is cyclic. Otherwise, go back to
5. After you have correctly implemented `ll_has_cycle()`, the program you get when y compile `ll_cycle.c` will tell you that `ll_has_cycle()` agrees with what the program it to output.

ACTION ITEM: Implement `ll_has_cycle()` and execute the following commands to mak that the provided tests pass.

```
$ gcc -std=c11 -Wall ll_cycle -o ./ll_cycle
$ ./ll_cycle
```

Hint: There are two common ways that students usually write this function. They differ they choose to encode the stopping criteria. If you do it one way, you'll have to account special case in the beginning. If you do it another way, you'll have some extra NULL che which is OK. *The previous 2 sentences are meant to urge you to not stress over cleanline they don't help you, just ignore them.* The point of this exercise is to make sure you know

res in

nm for

he

ave found

o step 2.
you
n expects

ke sure

in how
for a
ecks,
ess. If
w how to

use pointers.
Here's a [Wikipedia article] on the algorithm and why it works. Don't worry about it if you
completely understand it. We won't test you on this.
As a closing note, the story of the tortoise and the hare is always relevant. Writing your
programs slowly and steadily, using debugging programs like GDB, is what will win you

---
--
Credits: the exercise come from CS61C spring 2019

Recitation_7
slides

Linked Structures

•Self-referential structs can be used to create linked

```
struct node {
        int data;
        struct node *next;
};
typedef struct node node;
```

• **next** holds the address of a **node struct**

• through the next pointer we can link **node** structs

don’t

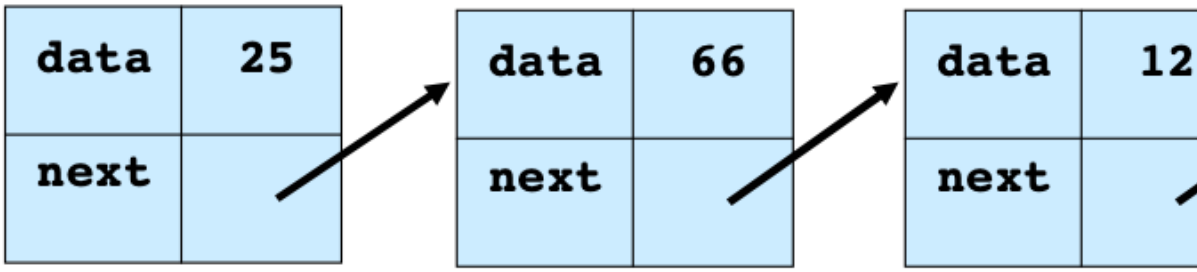
C
the race.

---

data structures:

together:

- through the next pointer we can link **node** structs

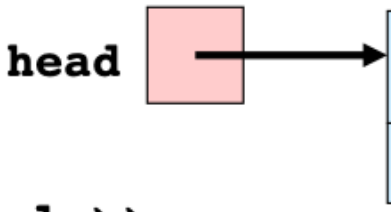


Linked List

- Ordered Collection of data
- Need a single variable which is pointer to 1st node
- nodes are linked together in-order by following **n**

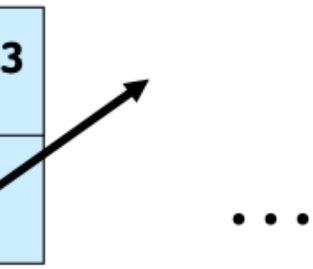
```
node *head;
head = malloc(sizeof(node));
head->data = 25;
head->next = NULL;

 head->next = malloc(sizeof(node));
 head->next->data = 99;
 head->next->next = NULL;
```

List of



List of leng

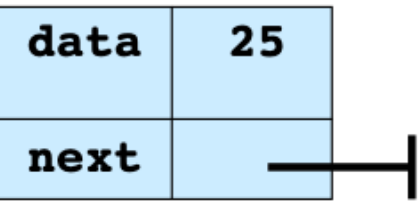
together:



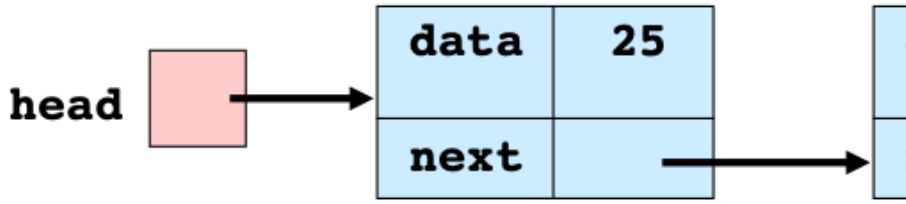
. . .

e on list
ext pointers

length 1:

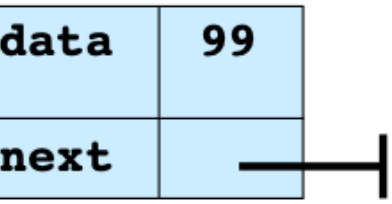


gth 2:



Operations on a List

- All start at node pointed to by head and traverse next pointers to access nodes in the list
- Accessing the ith node is O(n):
  - first access head node, follow its poi access the 2nd node, follow its pointe the 3rd node, and so on



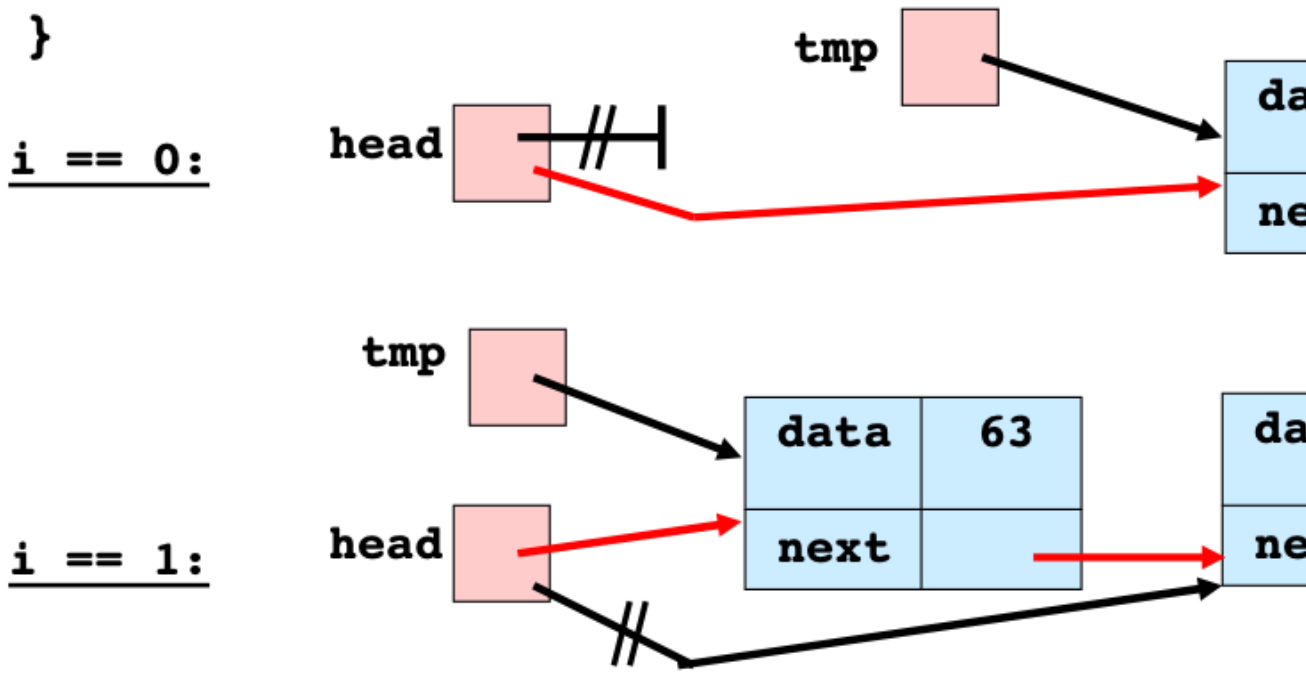
t

d pointer,
s other

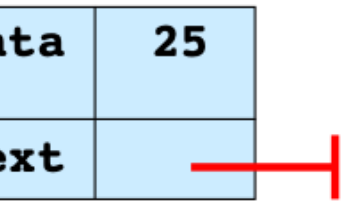
nter to
r to access

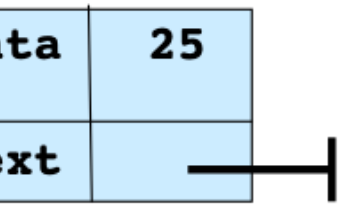
Insert at Head of List

```
head = NULL;
for(i=0; i < 10; i++) {
        tmp = malloc(sizeof(node));
        if(tmp == NULL) { Error("malloc fa
        tmp->data = GetNextDataValue();
        tmp->next = head;
        head = tmp;
}
```

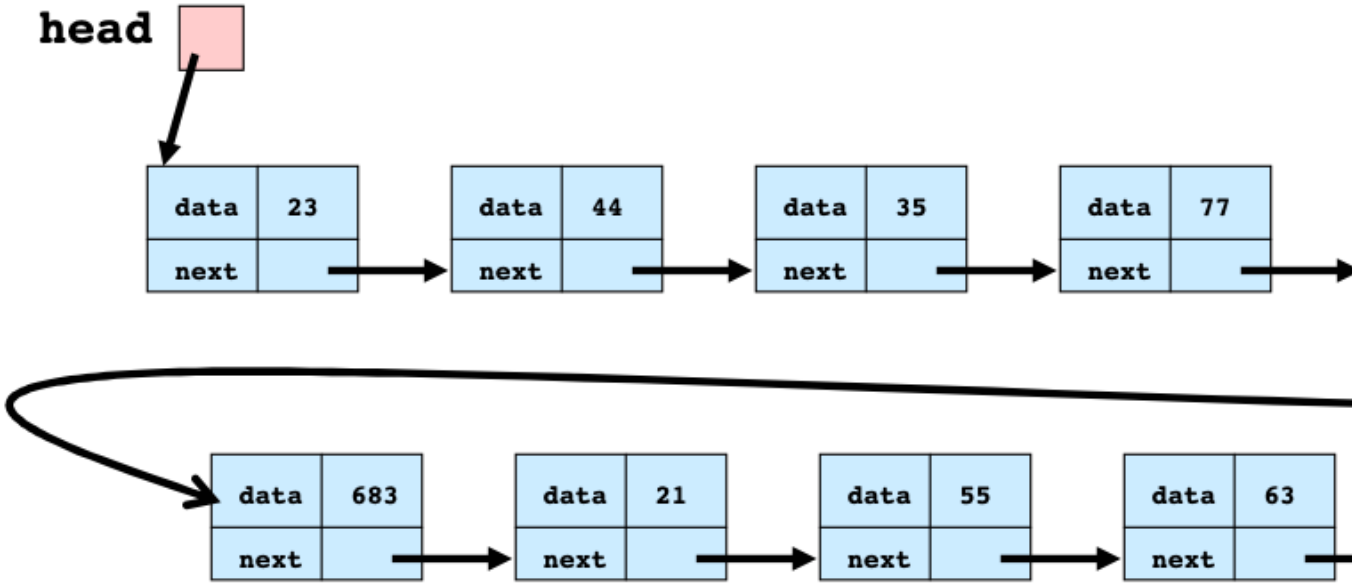


```
iled"); }
```

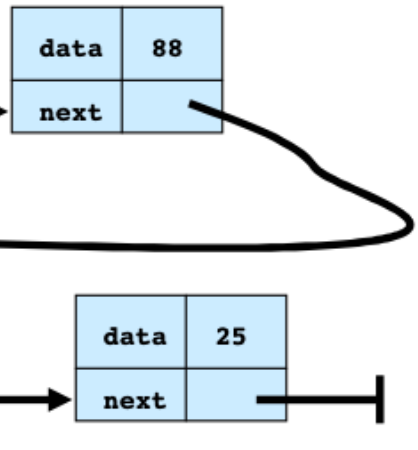




Resulting List of 10 no

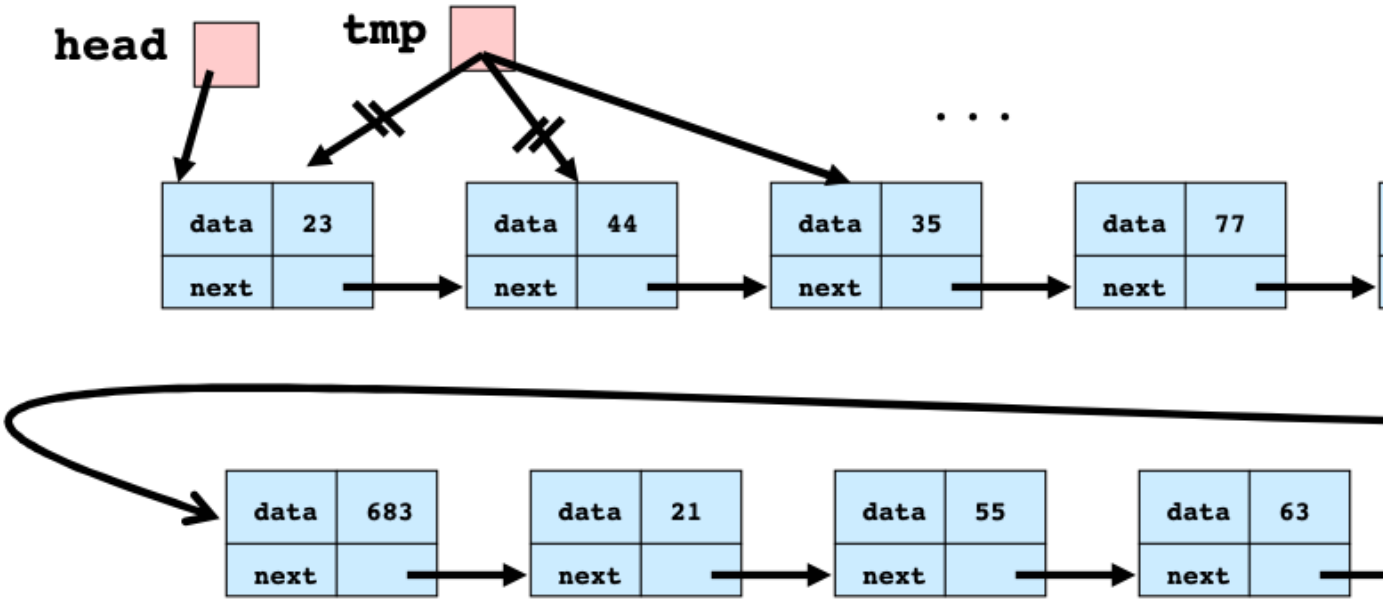


des:



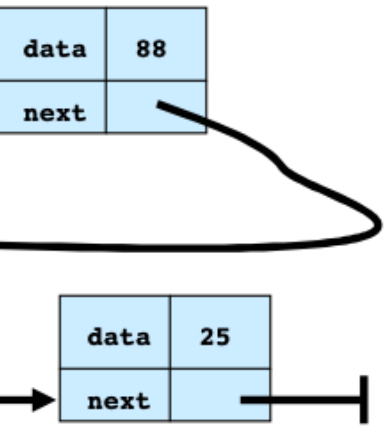
Traverse the List

```
tmp = head;     // start at the 1st node
while(tmp != NULL) {
        printf("%d ", tmp->data);
        tmp = tmp->next; // make tmp point
}
// output:  23 44 35 77 88 683 21 55 63 2
```



to next node

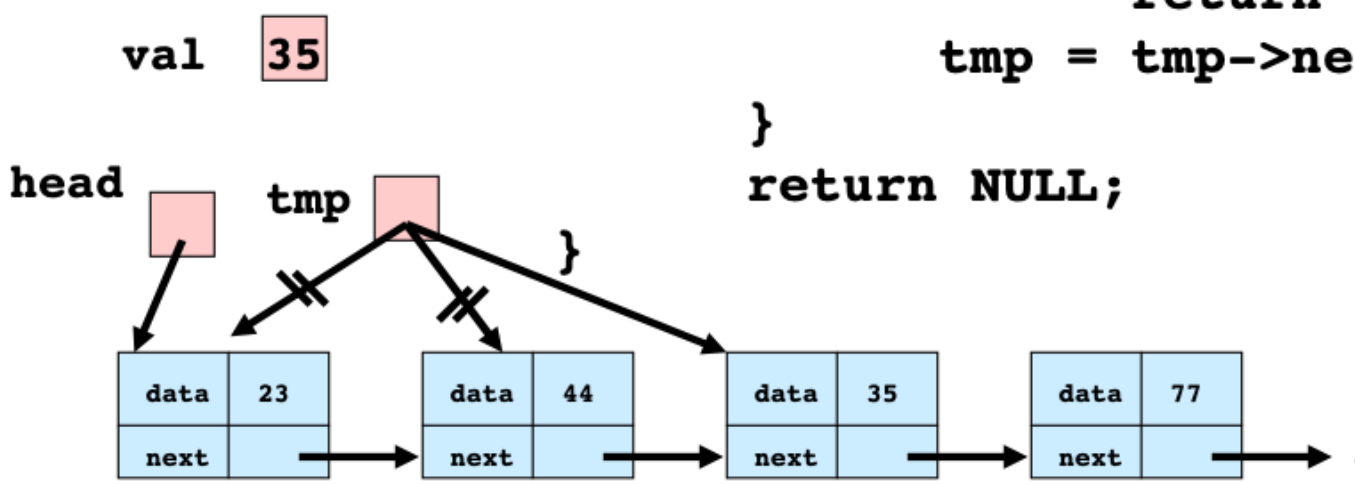
25



Find Element In List

- Start at head node, compare search value to data fi
- traverse next pointers until matching data field is f
  no more list

```
node *FindInList(node *he
        node *tmp;
        tmp = head;
        while(tmp != NULL)
                if(tmp->data
                        return
                tmp = tmp->ne
        }
        return NULL;
}
```

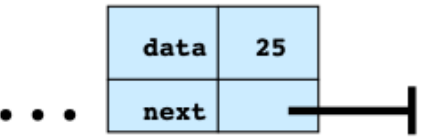


Insert in the middl

ield

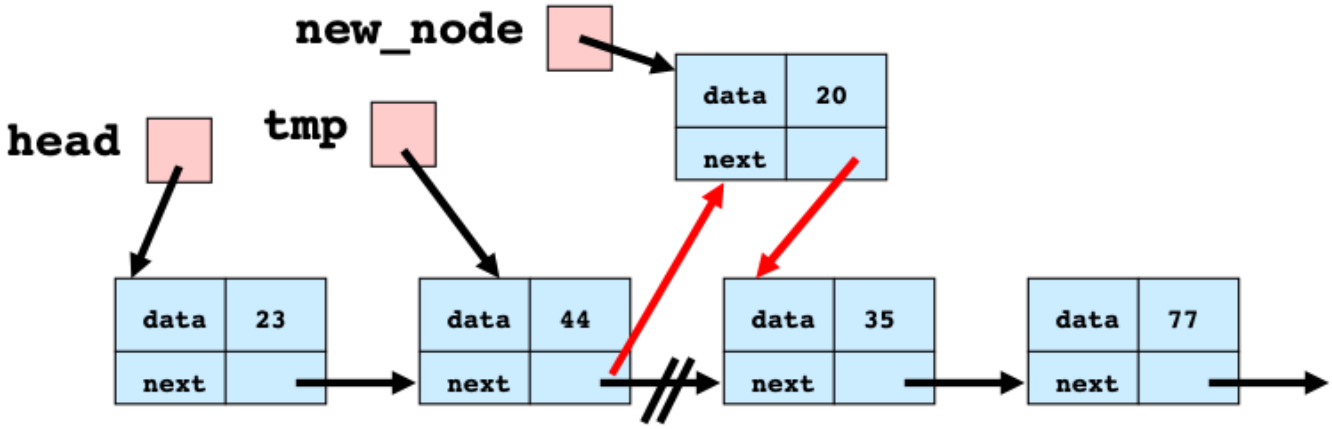
found, or until

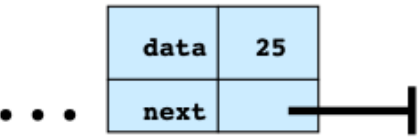
```
ad, int val) {

{
== val)
tmp;
xt;
```



e

```
node *new_node, *tmp, *head;

new_node = malloc(sizeof(node));
new_node->data = 20;
tmp = head->next;

// insert new_node after tmp
new_node->next = tmp->next;
tmp->next = new_node;
```




data
25
next