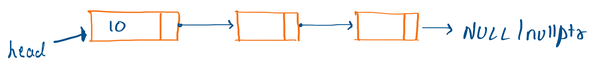
Day 5

Friday, 17 March 2023 9:07 PM

Linked List

- Sequential DS like arrays
- Elements don't need to be in contiguous location.



- Store data and pointer / reference
  - C++
  - Java
- No need to preallocate memory.
- Insertion / deletions can be done effectively at middle.

Code

```
struct Node {
    int data;
    Node *next;
    Node(int x) { data = x; next = NULL; }
};
```

⇒ Should know iterative as well as recursive solution of each problem

Problem: 1 → Middle of a Linked List

↳ leetcode.com/problems/middle-of-the-linked-list

↳ Naive Approach: ① Count Nodes (N)
② return [N/2]th element from Linked List.

↳ Efficient Approach: ① Slow and Fast ptr approach.

② slow = fast = NULL

movement ops.:
slow = slow → next
fast = fast → next → next

③ Stop when fast == NULL (Even Nodes)
or
fast → next == NULL (Odd Nodes)

④ output → slow → data

Pointers

→ Used to store address of a variable → ① Address of variable
② Reference to variable

→ & : reference operator
↳ when used before a variable name (while not declaring it) gives address of the variable

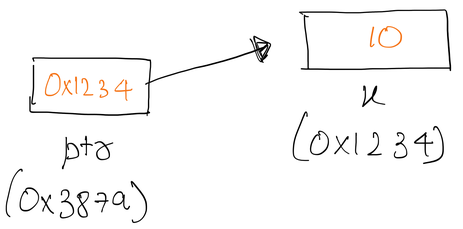
→ * : De-referencing operator / star operator → ① Value at an address
② Create pointer variable
↳ when used before an address (or address variable, i.e., pointer) gives the value of the address.

Code

```
int x = 10;
cout << (*(&x));   // 10
```

```
int x = 10
int *ptr = &x;
cout << *ptr ;   // 10
cout << ptr ;    // 0x1234
```

Variable	Value	Address
x	10	0x1234
ptr	0x1234	0x3879



Problem: 2 → Delete Node in a Linked List

↳ leetcode.com/problems/delete-node-in-a-linked-list

↳ Head is not given

↳ Node is not last node of LL.

Problem: 3-4 → Linked List Cycle

↳ leetcode.com/problems/linked-list-cycle

(Medium) → ↳ leetcode.com/problems/linked-list-cycle-ii

↳ Floyd's Cycle Detection Algorithm / Slow-Fast / 2 ptr Approach / tortoise and hare approach