COP3502 2/19/26

1) Final Project - command line compile

2) Linked Lists

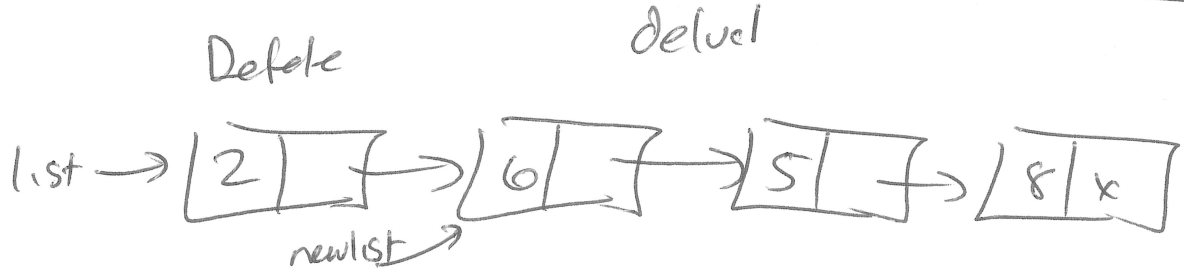
✓ - delete
✓ - reverse
} code these

- Doubly LL
- Circular LL
} talk, draw pics

3) Example

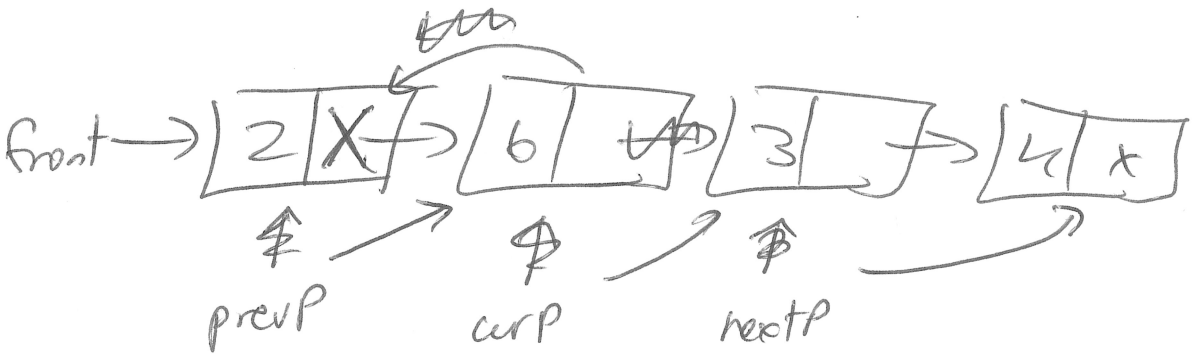
- updating a "dictionary" of words allowing add freq, delete remove lower freq, search

---



B.C.
- list NULL ret NULL
- 1st item to be delete
  - → save ptr 2nd item
  - → free (1st node)
  - → return saved ptr

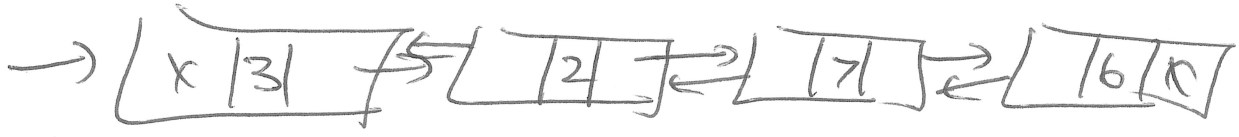
else list->next = remove (list->next, delval);
return list;



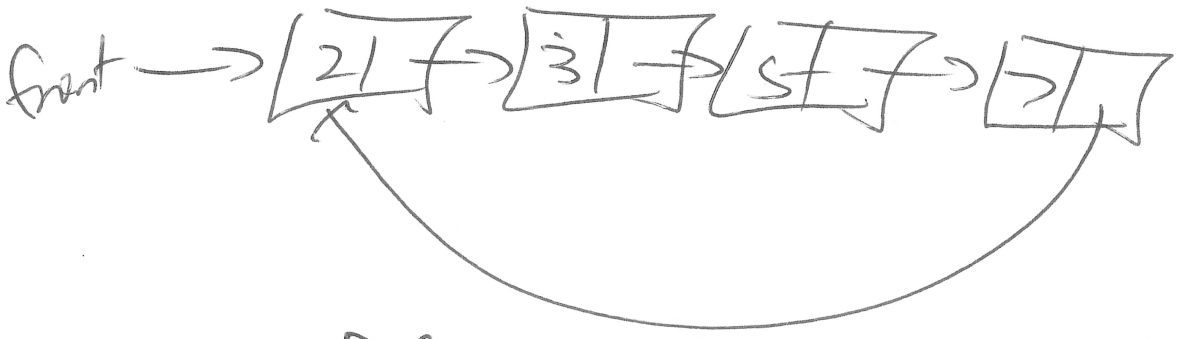
1. ~~prevP~~->curP->next = prevP
2. ~~nextP = nextP->next~~ (oops loses 3)
   prevP = curP
3. curP = nextP
4. nextP = nextP->next;

Doubly LL

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

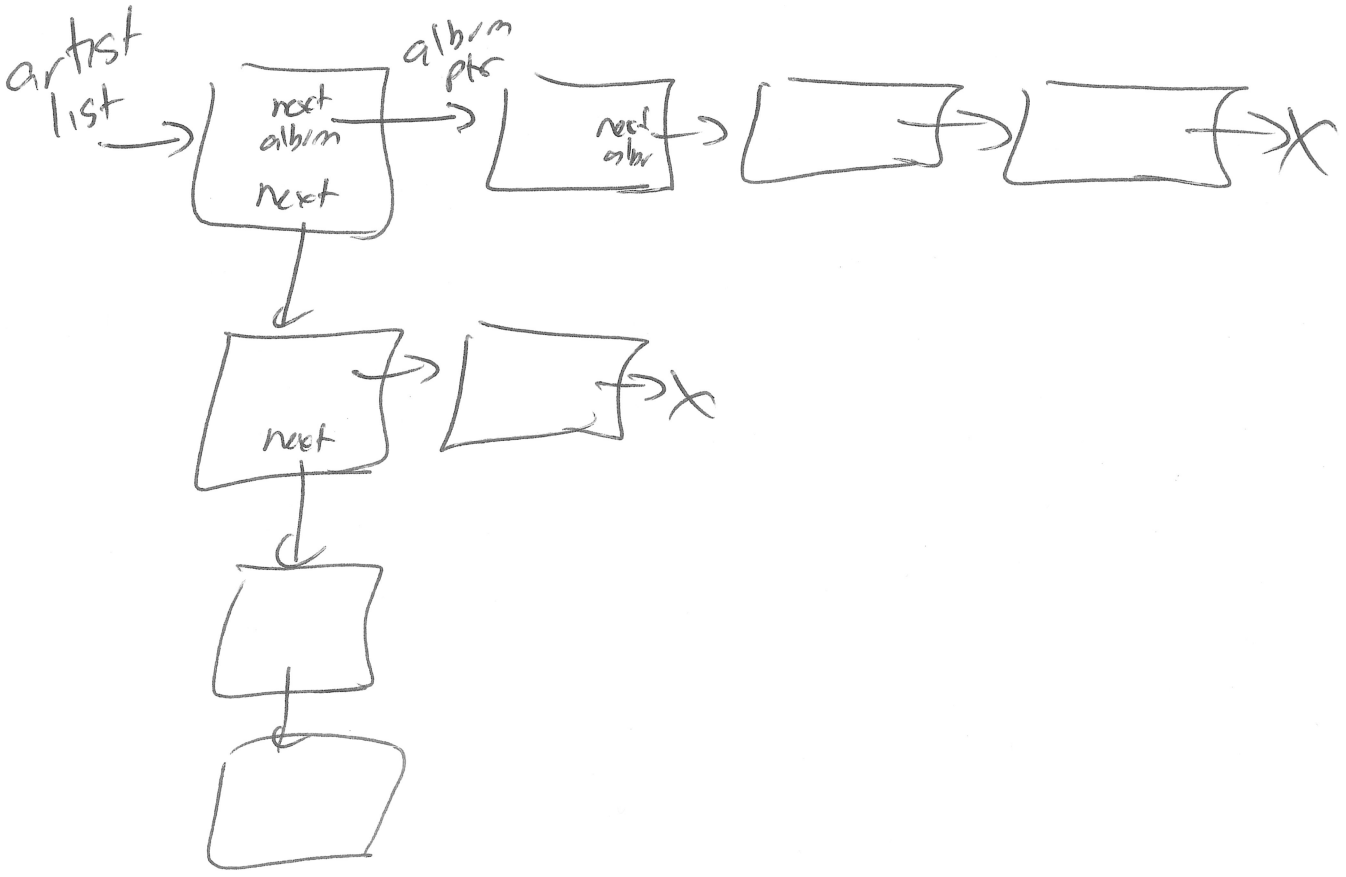


1) Benefit - you don't lose access by going -> next
2) Drawback - more storage
   MORE WORK TO MAINTAIN POINTERS



Differences in maintaining structure

Where's the end?



My example

"String" → "int"

aardvark → 10

ate → 100

functionality

+1 string

-1 String

~~Seq~~ get returns freq for string

remove

empty=0, a=1, b=2, .. z=26

cat = ~~3×26²~~ + $3\times27^2 + 1\times27^1 + 20\times27^0$

ba = $0\times27^2 + 2\times27^1 + 1\times27^0$