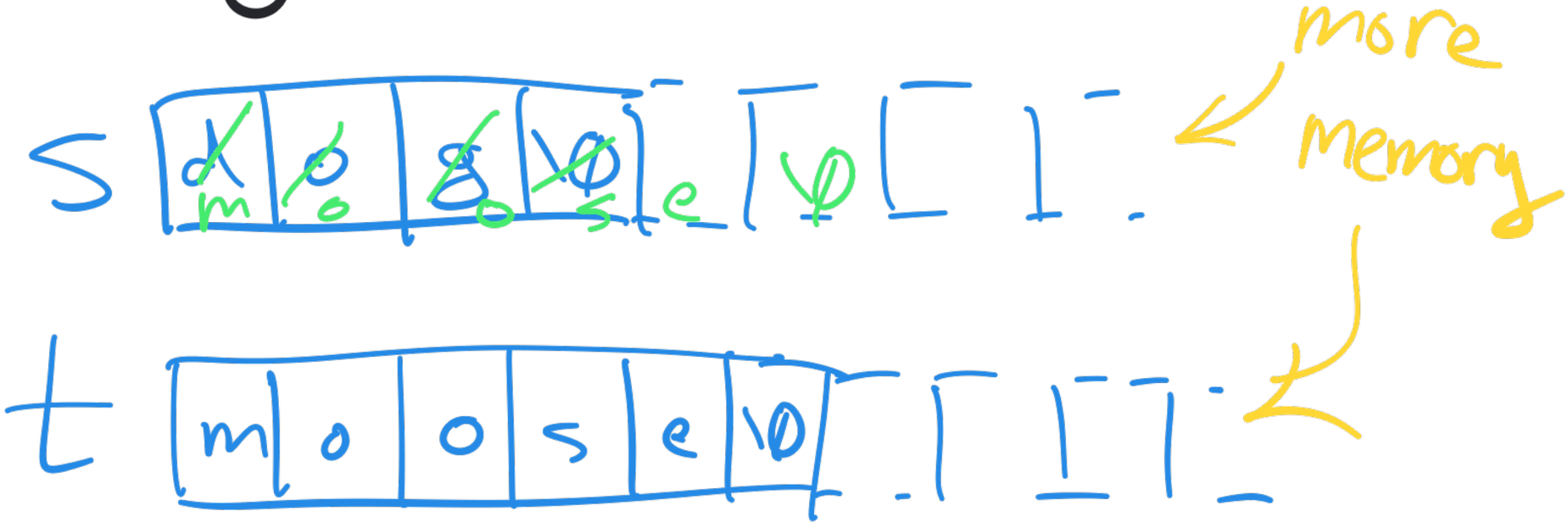
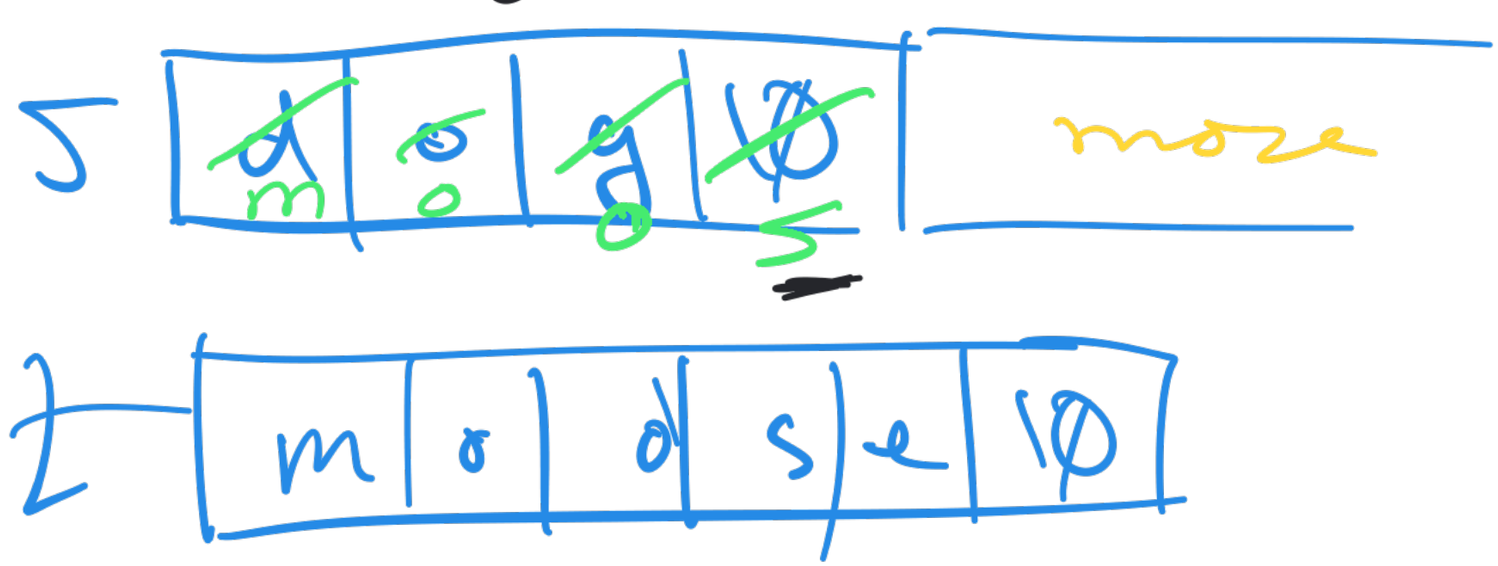
CS 208

Wed, 5 April 2023

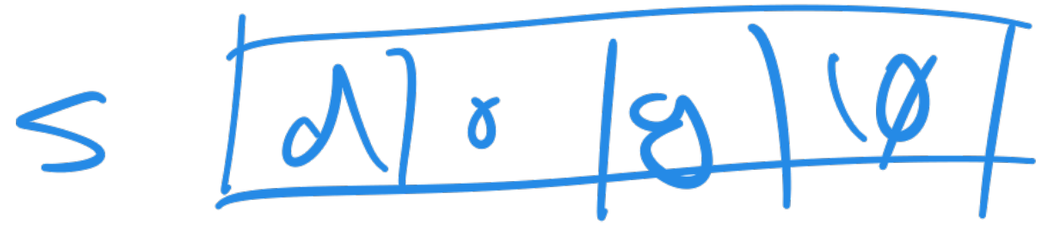
```
char s[4] = "dog";
char t[6] = "moose";
strcpy(s, t);
```

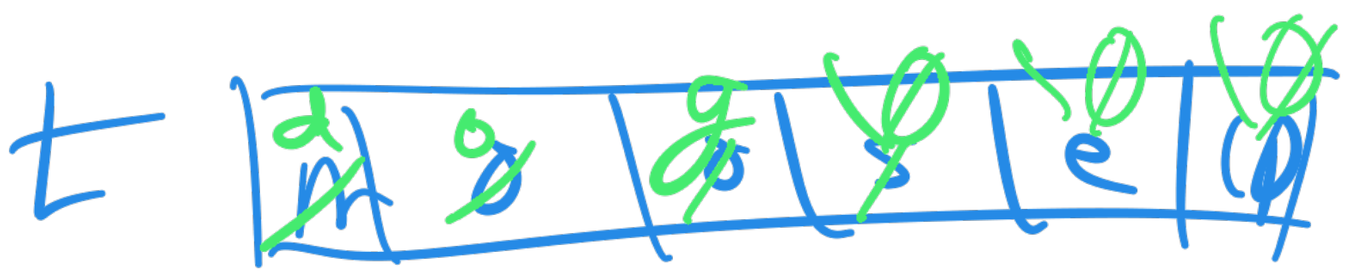


```
char s[4] = "dog";
char t[6] = "moose";
strncpy(s, t, 4);
```



```
char s[4] = "dog";
char t[6] = "moose";
strncpy(t, s, 6);
```





Here's a thing to do.

① strncpy(dest, src, dest_size);
dest[dest_size-1] = '\0';

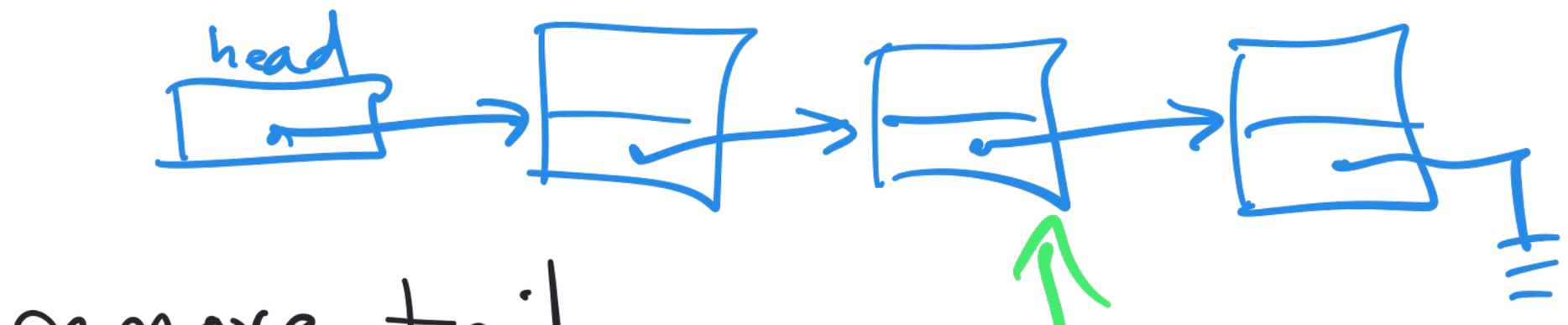
② Allocate dest memory based on src length

```
char *p = malloc(sizeof(char) *(strlen(src)+1));
if(p != NULL){
    strcpy(p, src);
}
```

type of NULL : void *

type of '\0' : char

Singly-linked list, only head



remove tail

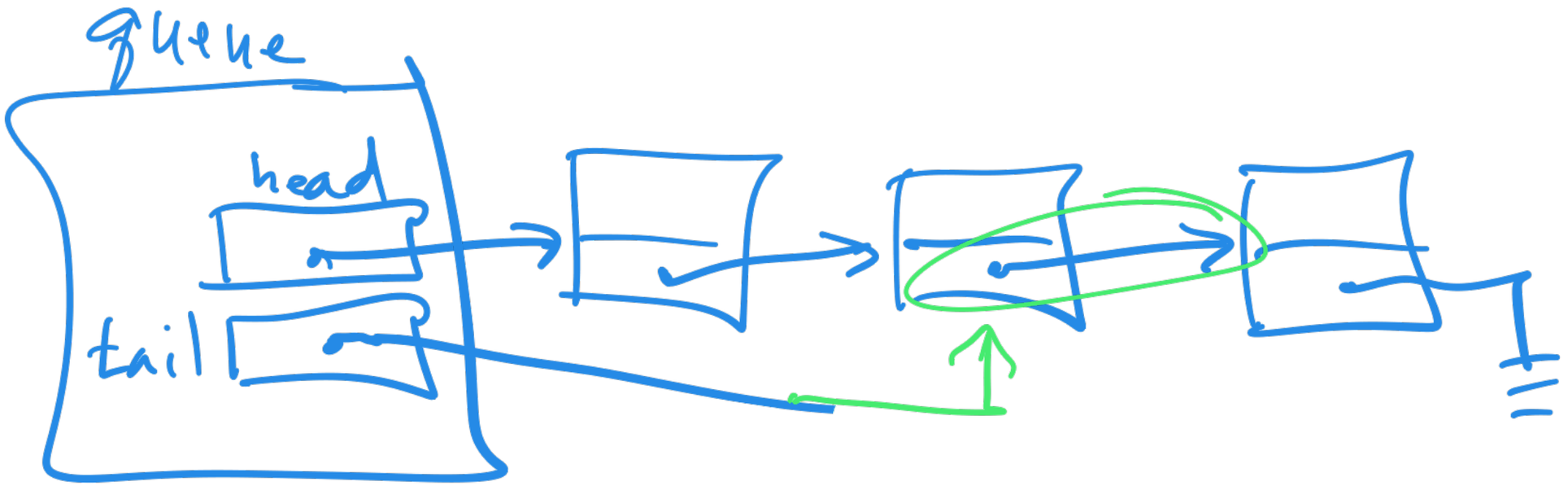
walk to here

- remove next node.

O(N)

operation

Singly-linked list, only head



remove from tail

WE'RE NOT DOING THIS

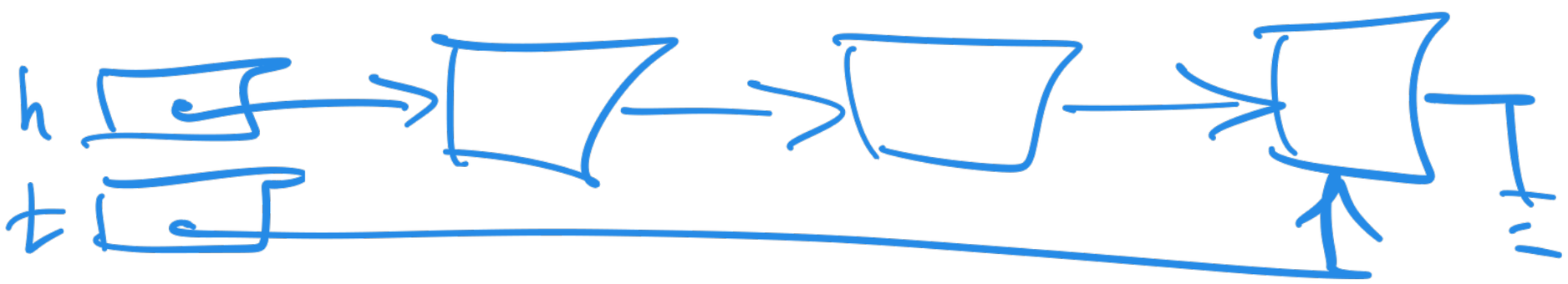
ARE DOING

add to head ←

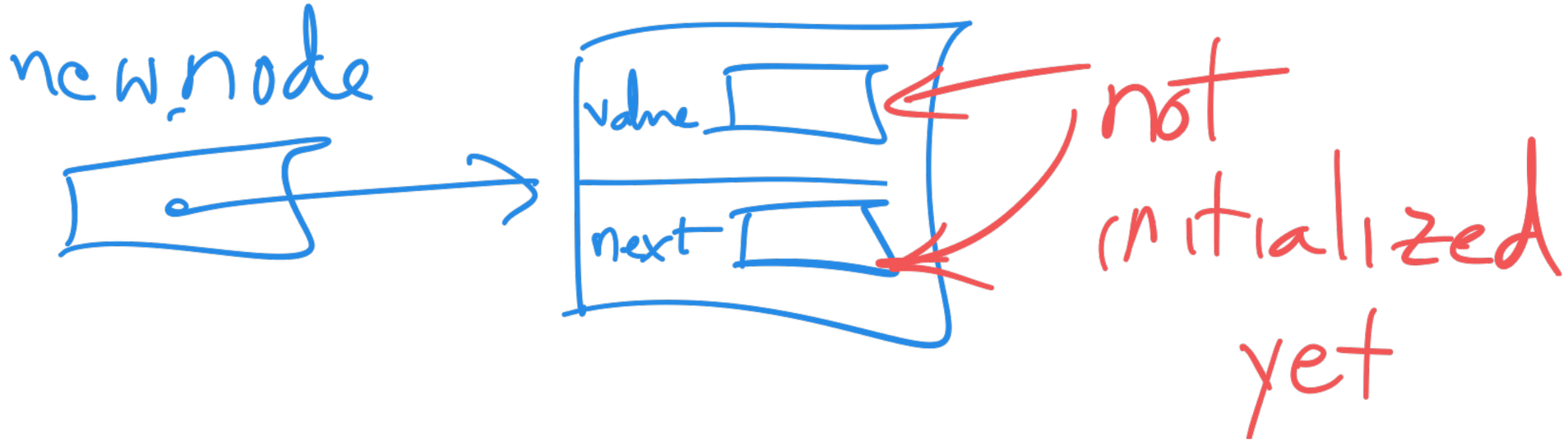
add to tail

remove from head

~~remove from tail~~



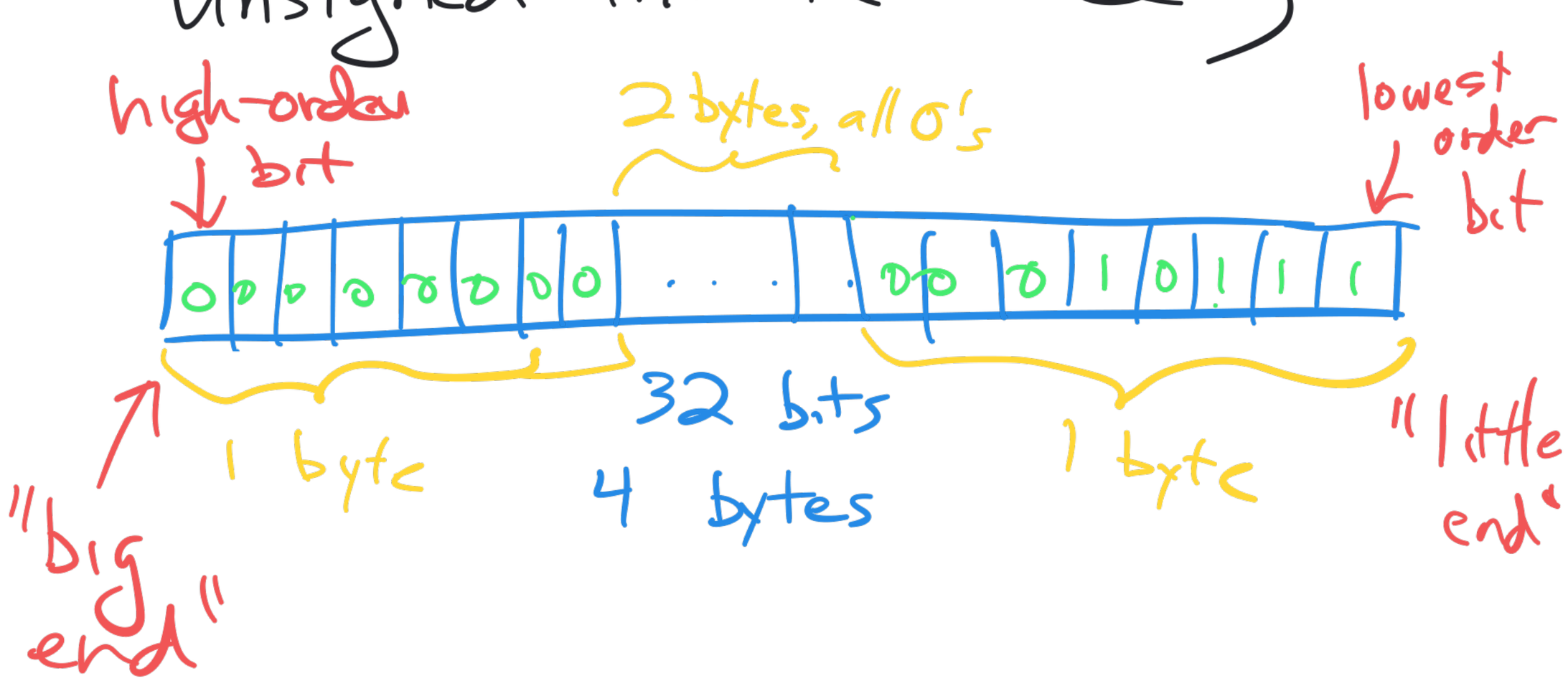
```
list_ele_t *new_node
        = malloc(sizeof(list_ele_t));
```



Integers in bits

unsigned int (C type)

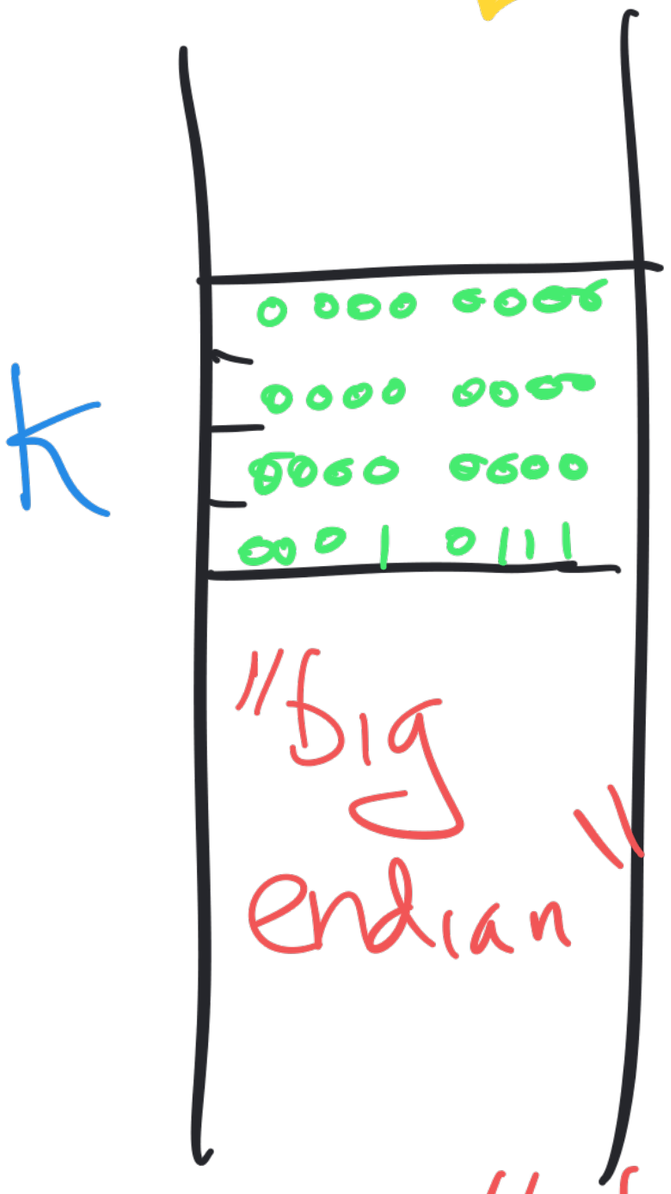
unsigned int k = 23; 16+4+2+1



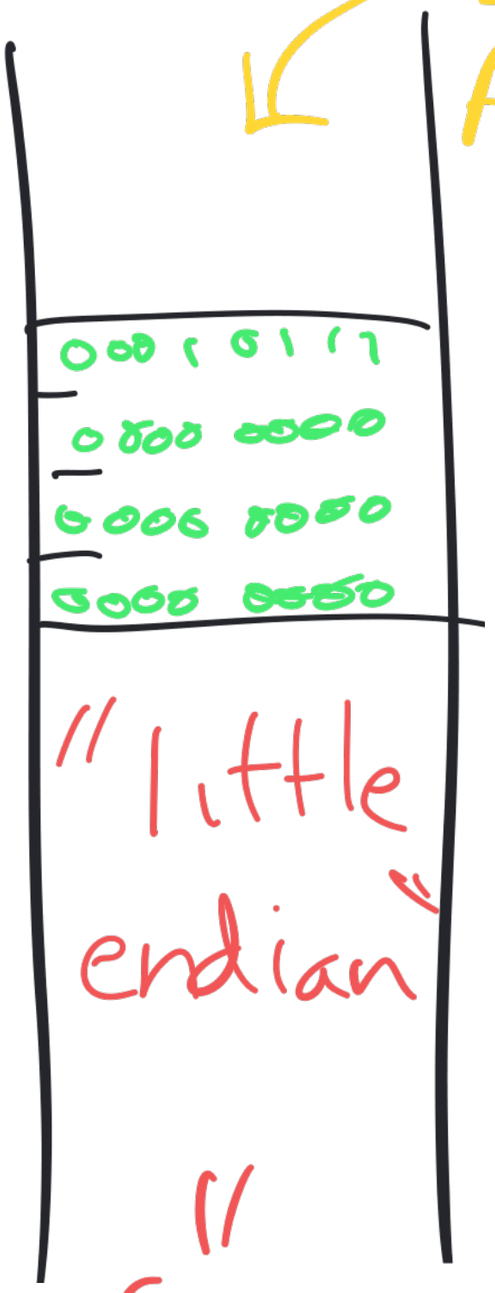
But wait

Motorola
ARM
MIPS

Intel
AMD



"big endian"



"little endian"

"byte order"

Negative integers

Pretend that our ints are 4 bits

(not 32)

Possible bit patterns:

unsigned

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

Negative integers

Pretend that our ints are 4 bits
=
(not 32)

Possible bit patterns:

"signed magnitude"

+

0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	-0
1001	-1
1010	-2
1011	-3
1100	-4
1101	-5
1110	-6
1111	-7

next:

"two's complement" integer representation