4/15.

Today

1) P5
2) Wrap up Heap.

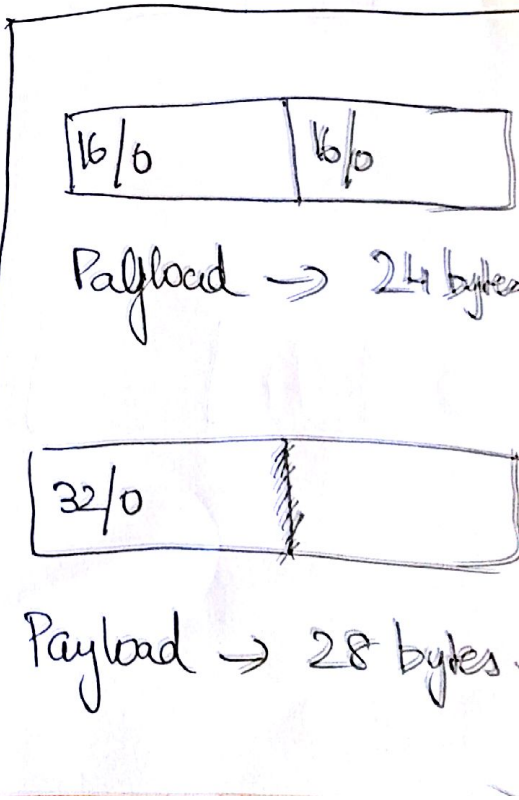
Last Class

1. Placement Policy
2. Splitting the block
3. Coalescing Free blocks.

↳ False Fragmentation.

↓ Fix ?

Merge the adjacent free blocks!

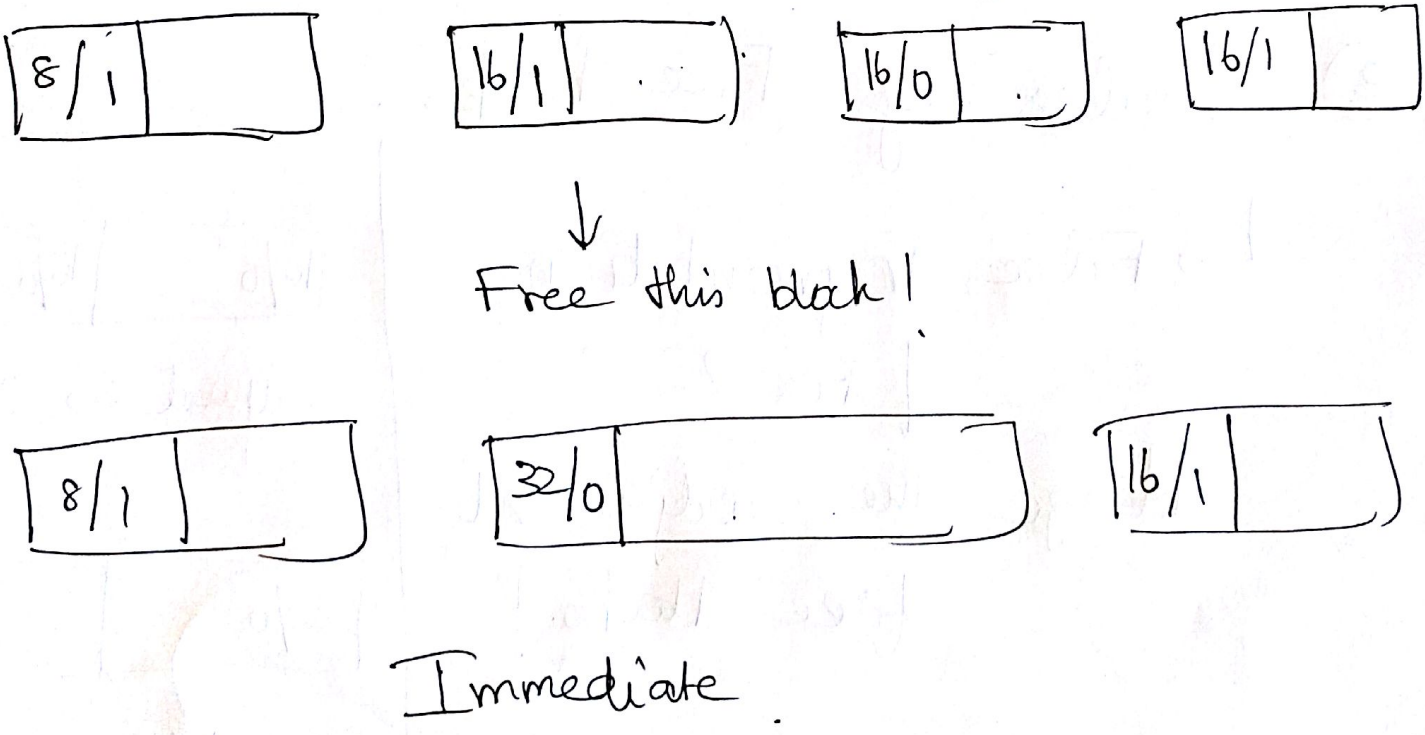


Scanned by CamScanner

When ?

① Immediately (Immediate Coalescion)
Merge any adjacent free blocks each time a block is freed.

② Deferred
"Do it later"



Immediate

Scanned by CamScanner

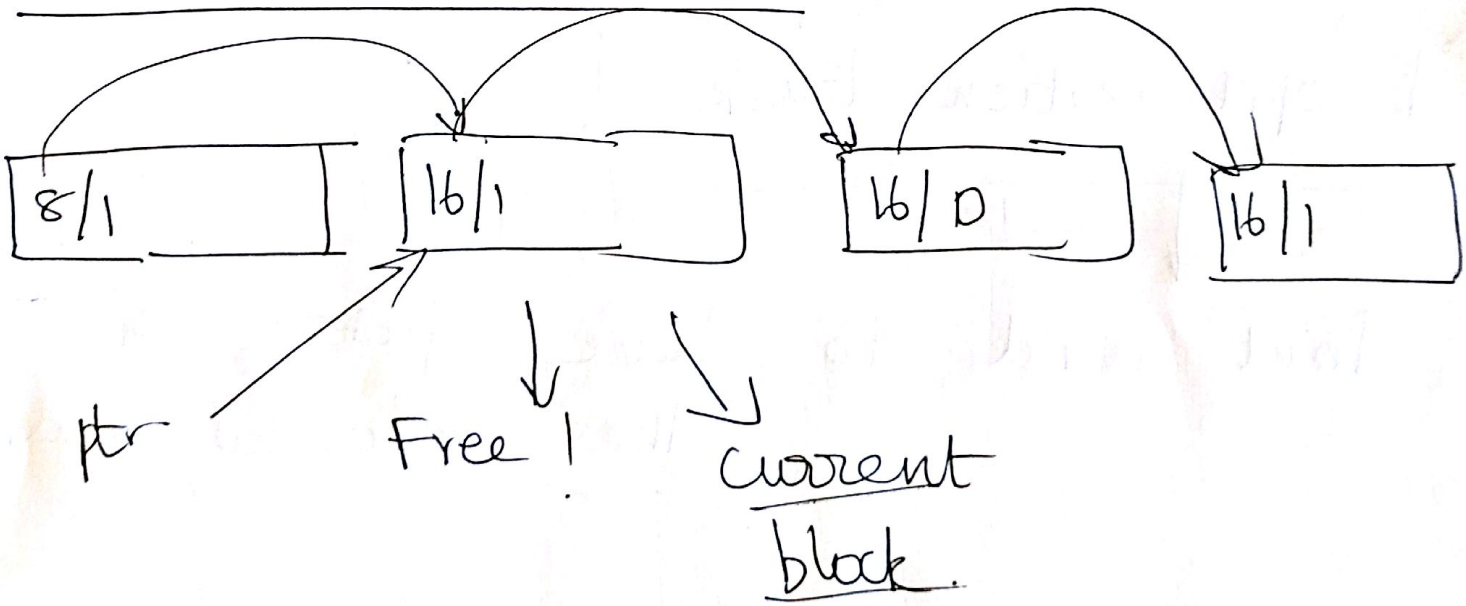
BUT !

Sequence of 3 word allocation requests and free requests.

Deferred might be better!

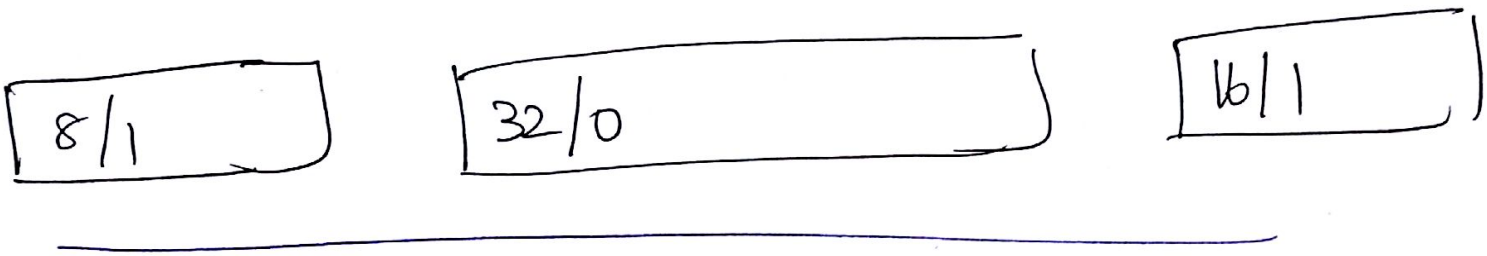
P5 → Immediate!

How to implement this? free(ptr)



→ check the next block's header and see if its free (and add the block sizes)

Scanned by CamScanner

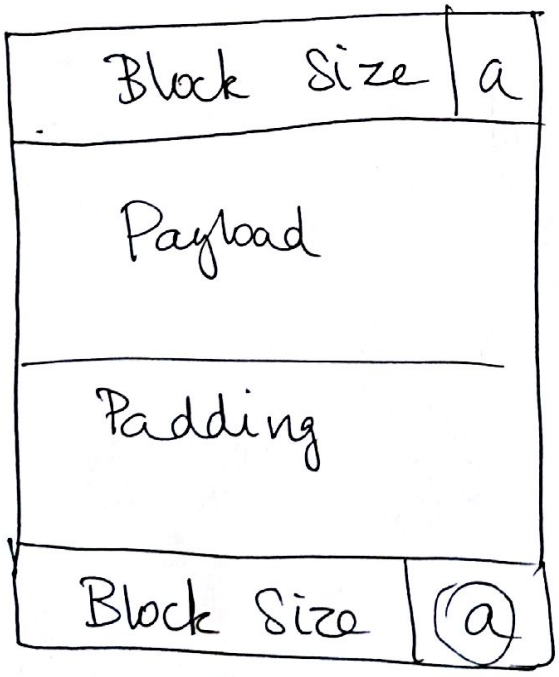


What if the previous block is free?
How to check that?

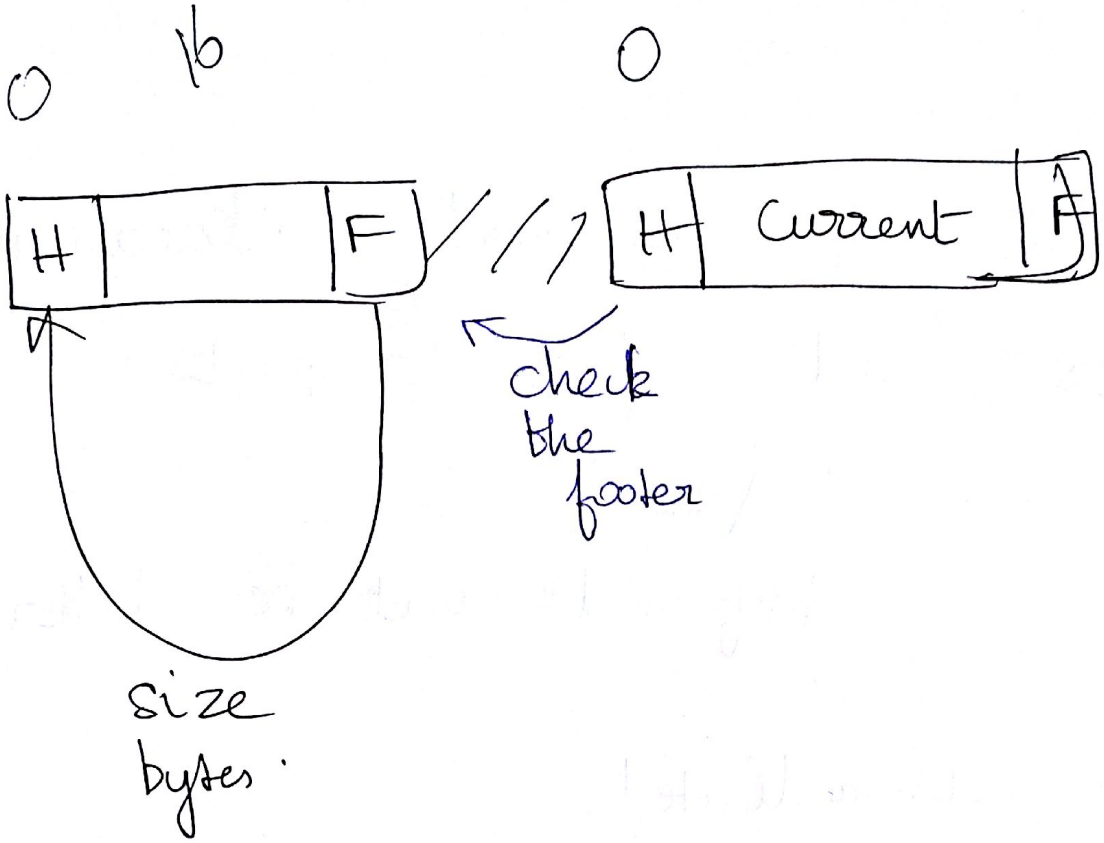
We need to traverse from the beginning! Expensive.

Add a footer! → replica of the header.

→ also called a boundary tag



Scanned by CamScanner



Drawback → Extra memory!

1 optimization trick!

↓

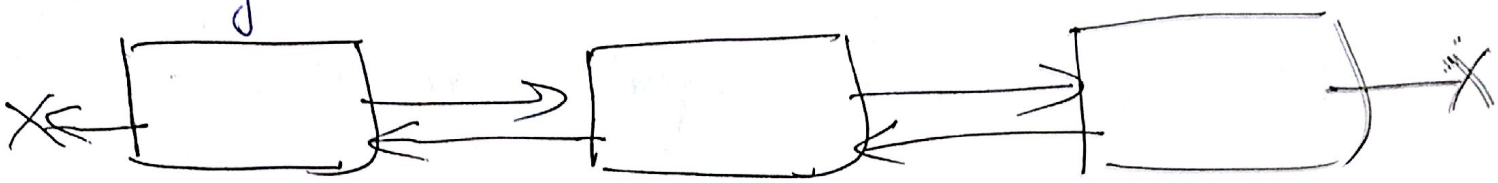
Dont need to have footers on the allocated blocks!

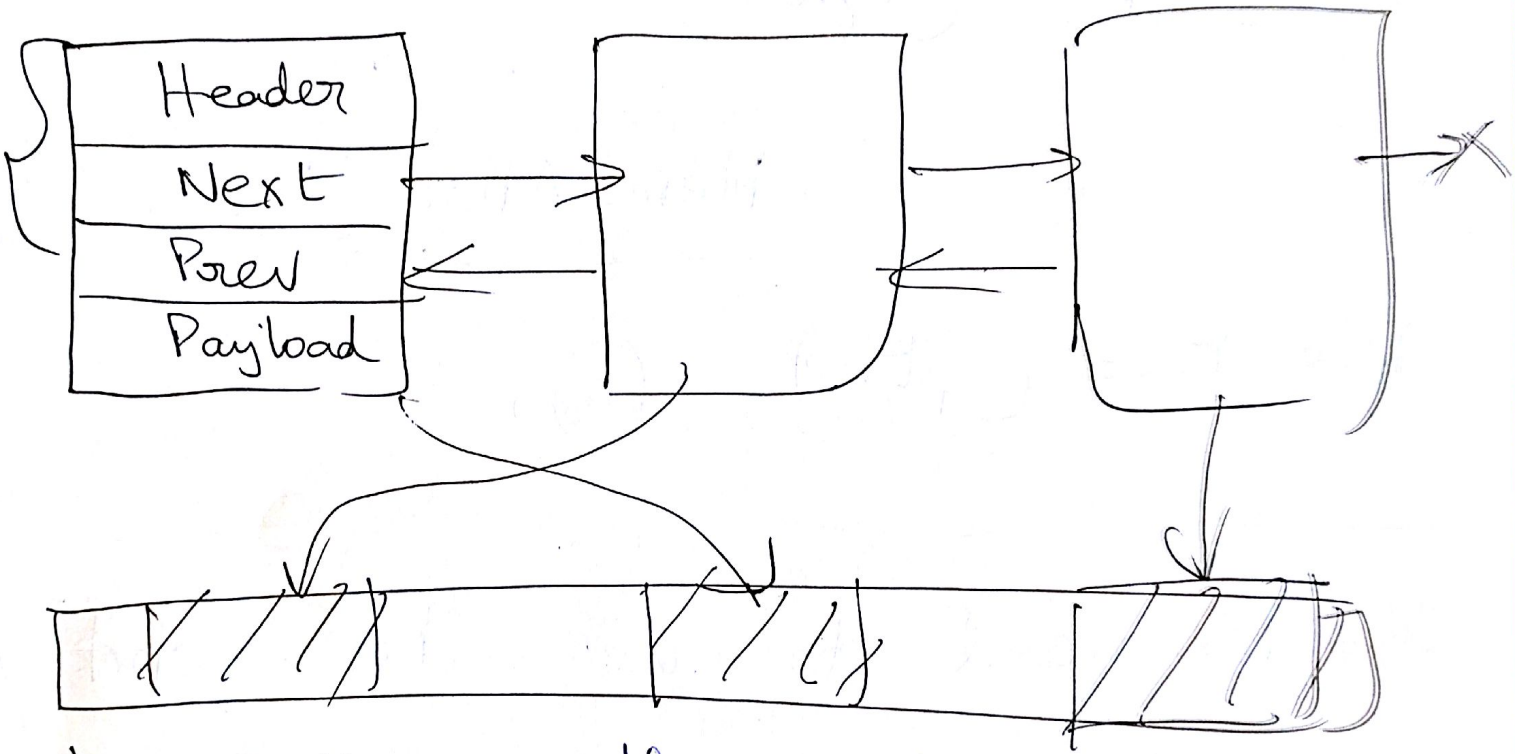
Pg. 826 in CS: APP

Scanned by CamScanner

Explicitly maintaining a Free List

Doubly linked List





Physically → they can be anywhere on the heap

Explicit Free List

↓

only free blocks!

First Fit → Instantaneous.

Scanned by CamScanner

P5

Sample.c → import "mem.h"

mem.c

MEM_INIT (bytes.) ✓

ptr = ( ) Mem_Alloc (bytes) ○

Mem_Free (ptr) ○

→ No need to use brk /sbrk / mmap

Single-word aligned!

↳ payload needs to start at an address that's a multiple of 4.

↓

start at any word!

Scanned by CamScanner

All blocks' size is a multiple of 4. ⟹

4	→	1 00	Encode allocation status (0 a)
8	→	1000	
12	→	1100	

Mem_Alloc (size) → Allocate

→ return a pointer to the payload.

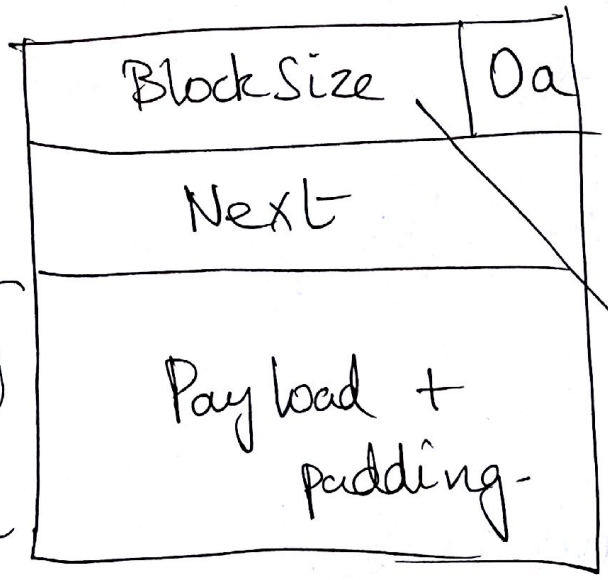
Header → Block size (+ allocation status)

+

Next pointer.

Total size of header = 8 bytes (2 words)

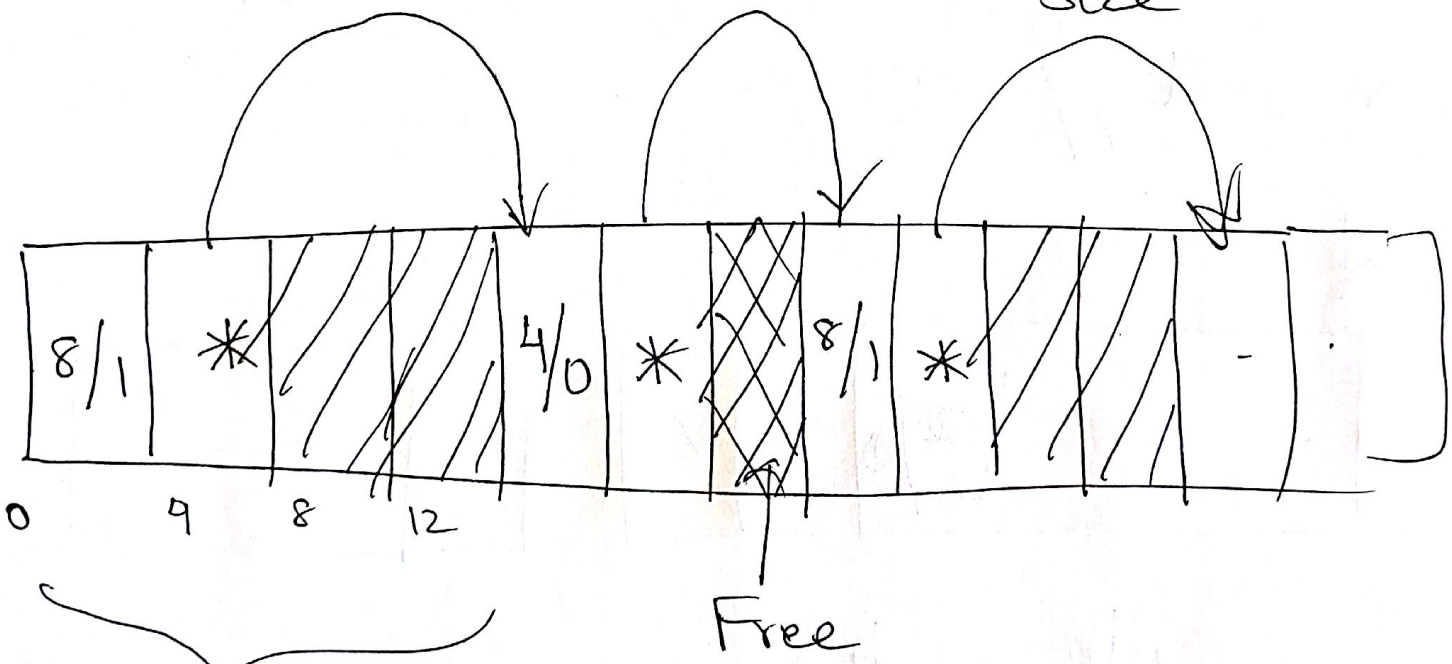
Scanned by CamScanner



} Header.

Only the (payload + padding) size.

Block size!



16 bytes

↓

8 → Header

8 → Payload

Scanned by CamScanner

Mem_Alloc (size)

① Round up!

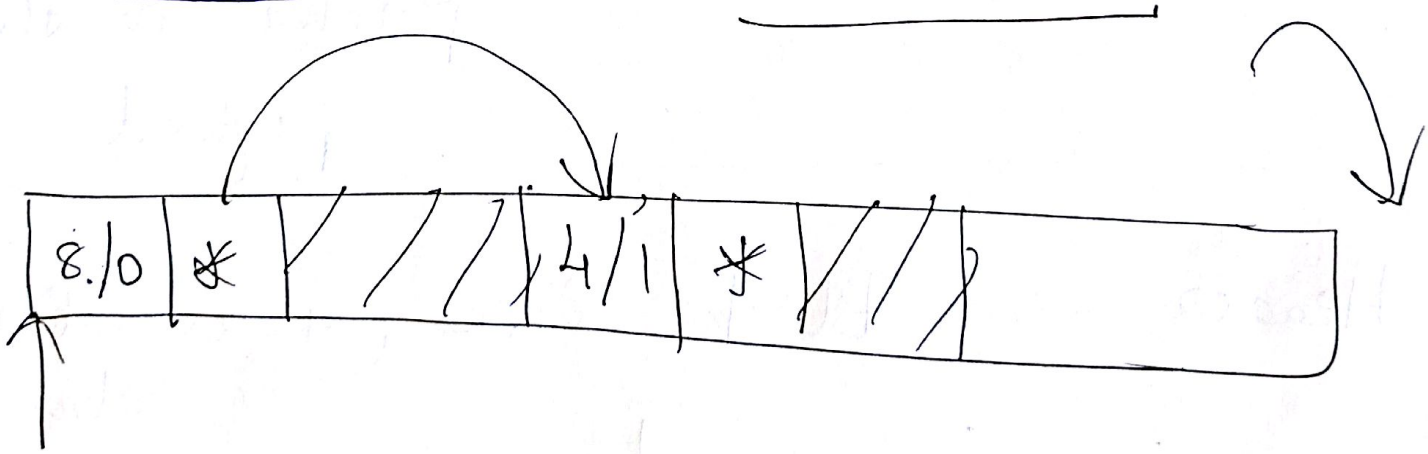
② Traverse the list to find the block thats the best fit.

Next Pointer

null at beginning.

ptr runner

ptr best_fit



global variable (list_head)

Scanned by CamScanner

runner is at end

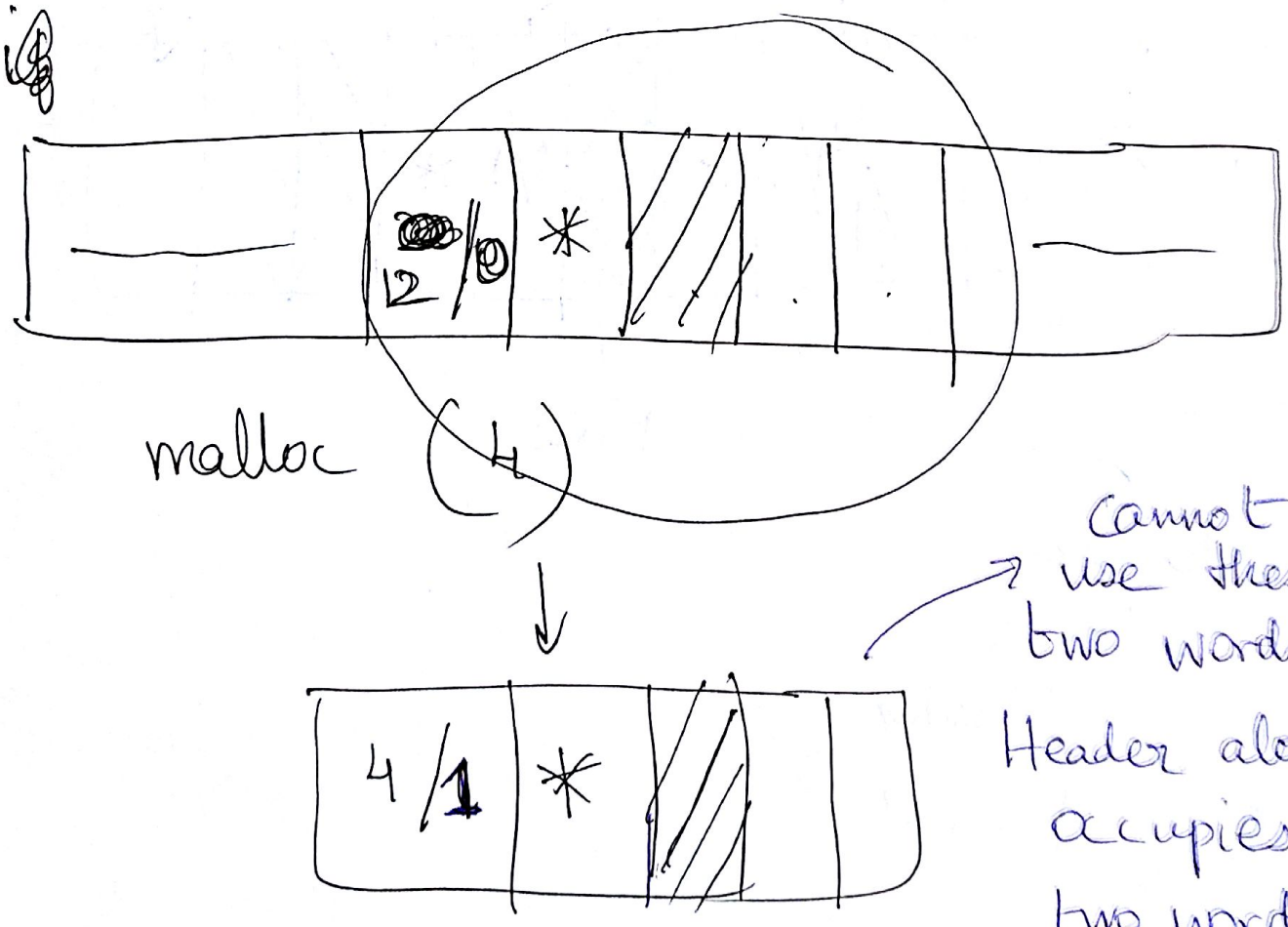
You check your best_fit //

↳ null ?

or not

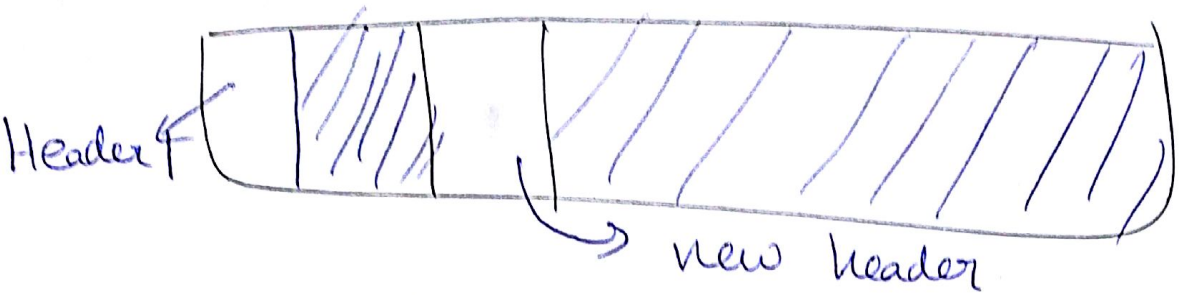
// Best_fit we have one !

check if we need to split it or not //



Scanned by CamScanner

If we do end up splitting,



↓

→ create a new header

→ update the next pointers

→ Don't forget to set the allocated bit

return the ptr to the payload

// End of Mem_Alloc

Scanned by CamScanner