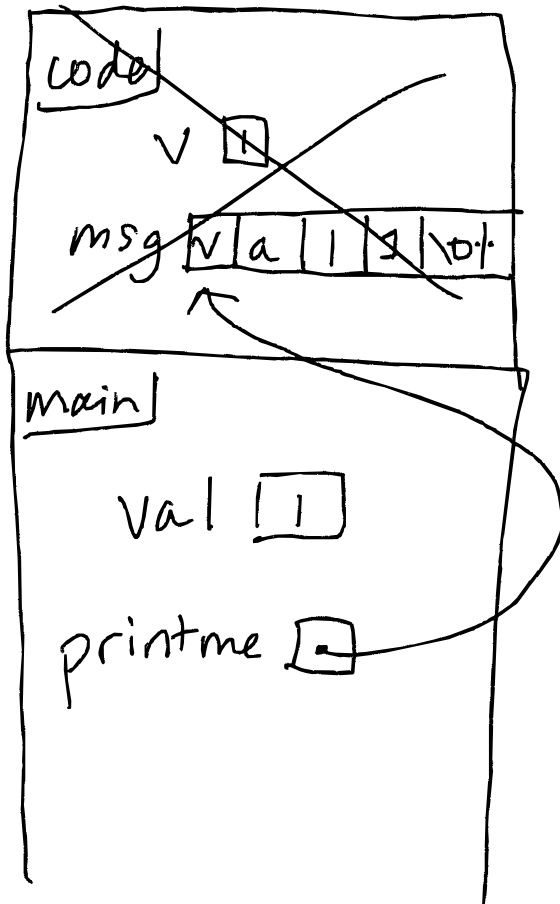


Return value gotchas #1

not safe x



read-only / static

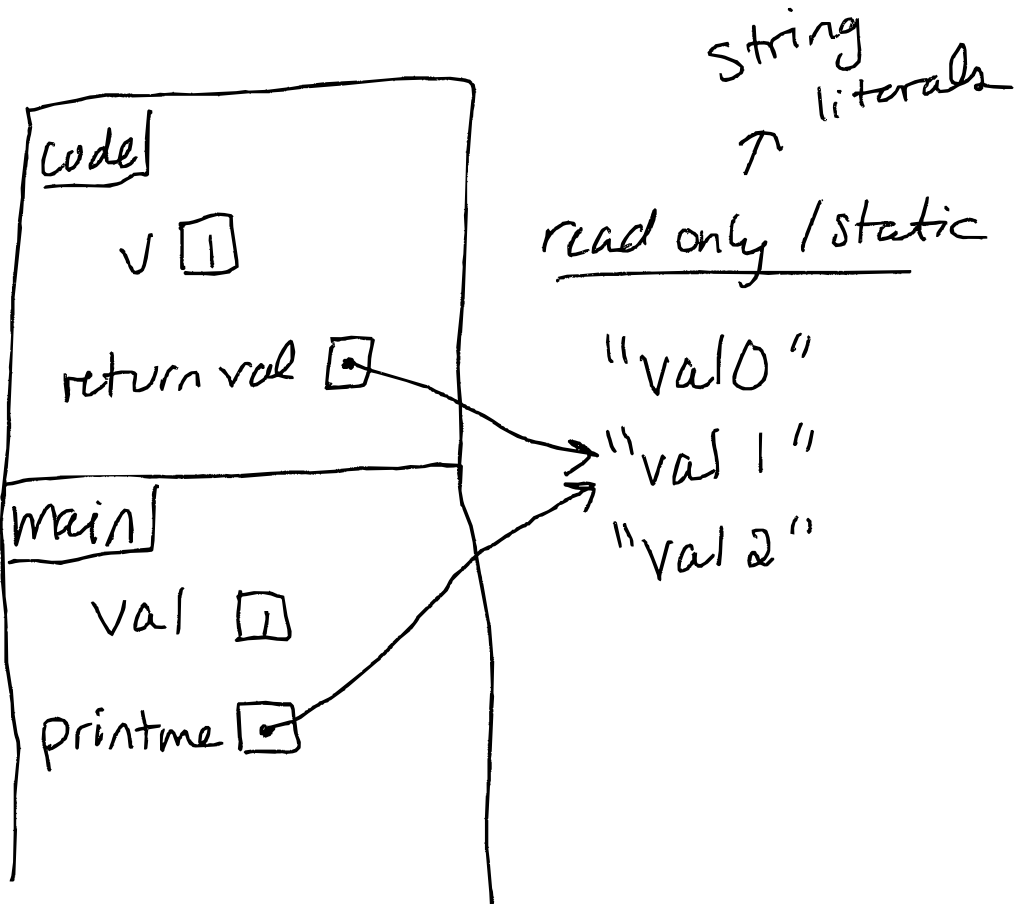
"val0"

"val1"

"val2"

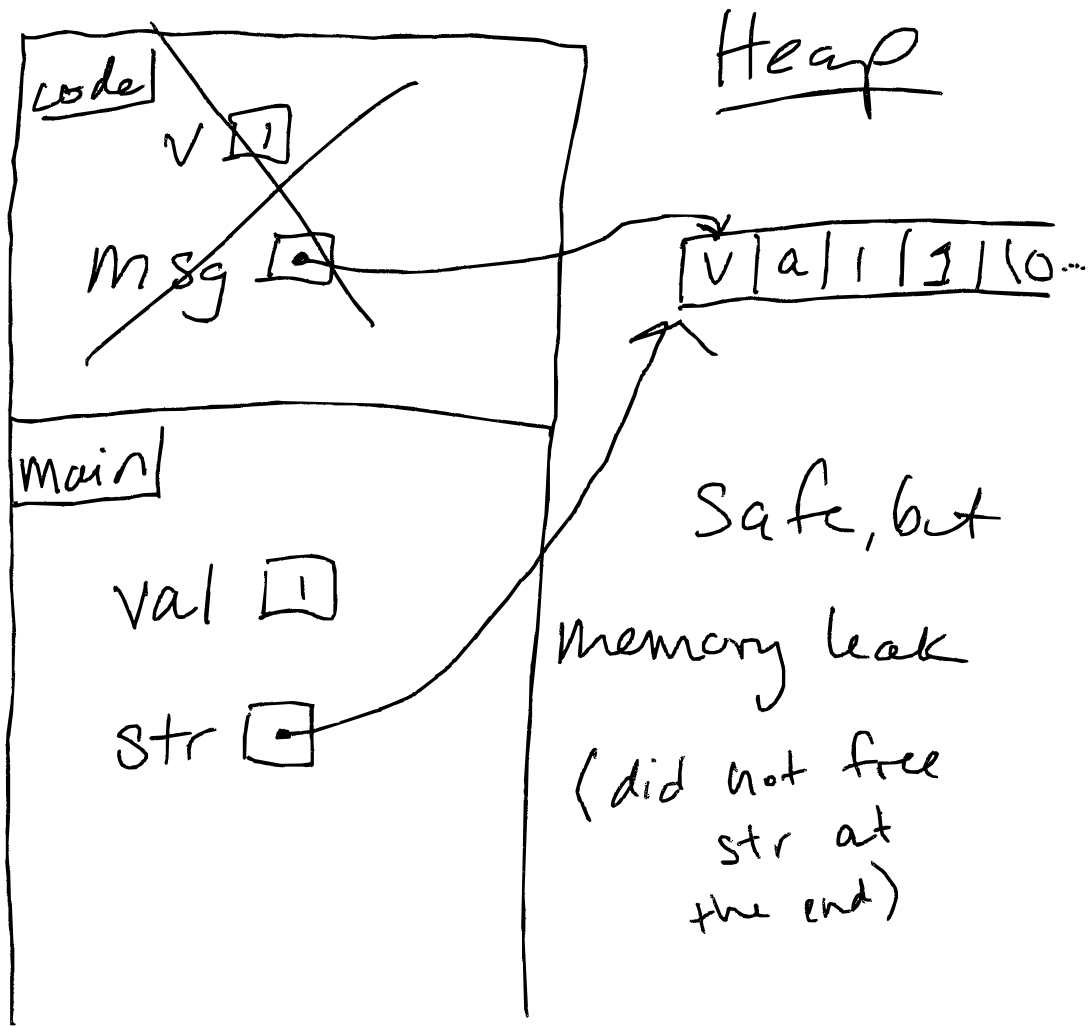
Return value gotchas #2

Safe ✓



Return value gotchas

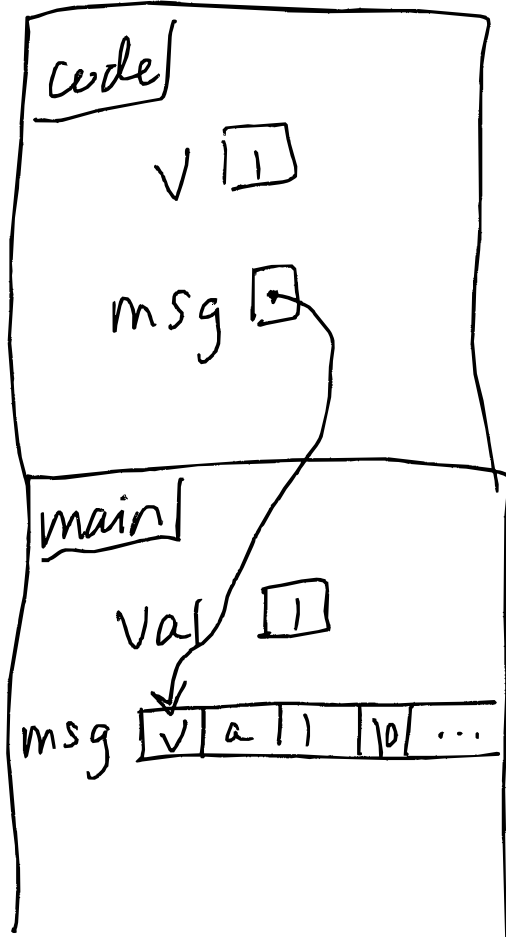
3



Return value gotchas

#4

safe ✓



passing a
ptr (b/c
array)

Standard I/O

- 1) `./a.out < infile.txt`
↳ contents of `infile` redirected to `stdin` of `./a.out`
- 2) `./a.out > outfile.txt`
↳ redirects `stdout` to `outfile`
- 3) `./a.out & > allfile.txt`
↳ redirects both `stdout` + `stderr`
- 4) `./a.out 2> errfile.txt`
↳ redirects `stderr`
- 5) `./a.out 1 > outfile.txt`
↳ same: `./a.out > outfile.txt`

Reading Text Files

fopen

num ☒ 1

sum ☒ 99

line

9	9	\n	\0	
---	---	----	----	------

grades.txt

99
79
51
92
56
⋮

fgets → gets the line

sscanf → gets the grade

atoi

Helpful String Commands

Tokenizing is the process of breaking a string into substrings (tokens)

ex : "2,4,6" → "2" "4" "6"

Strtok

each call to strtok finds the next delimiter, and replaces it with \0

So starting: "cat bear dog"

1st iter: "cat\0 bear dog"

new token 

list[0] → "cat"

then: strtok(NULL, delims) → means get NEXT delimiter

so "cat\0 bear dog"

token 

then list[1] → "bear"

⋮

and list[2] → "dog"

Linked List Example

struct node {

int data;

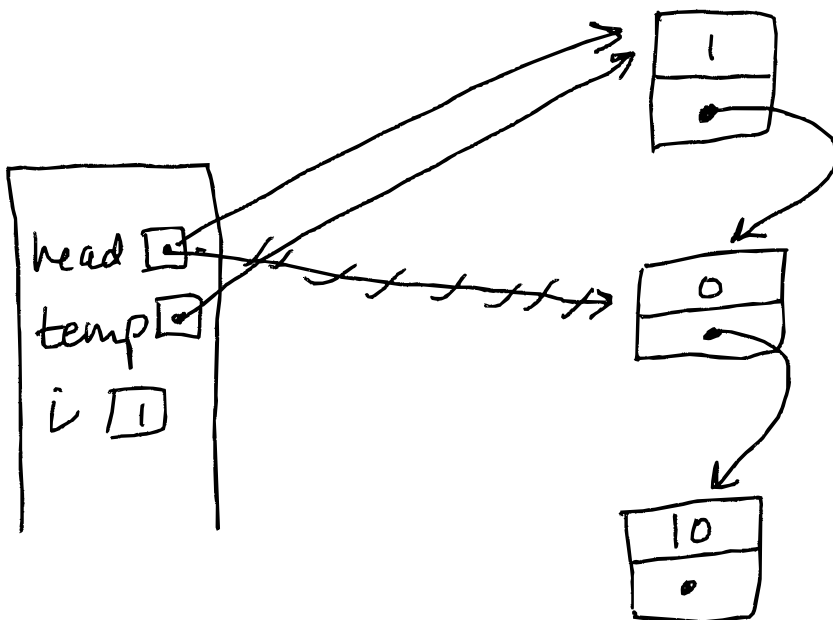
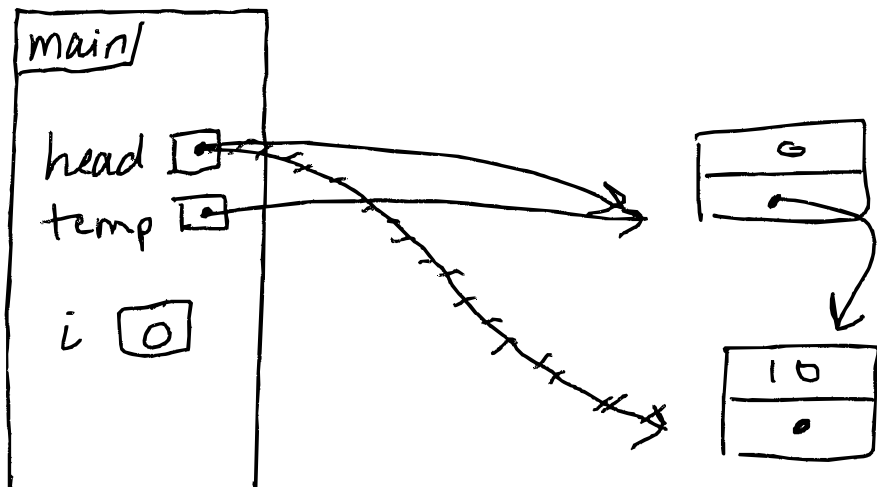
struct node* next;

}

size

4 bytes (int)

8 bytes (ptr)



Linked List in Java

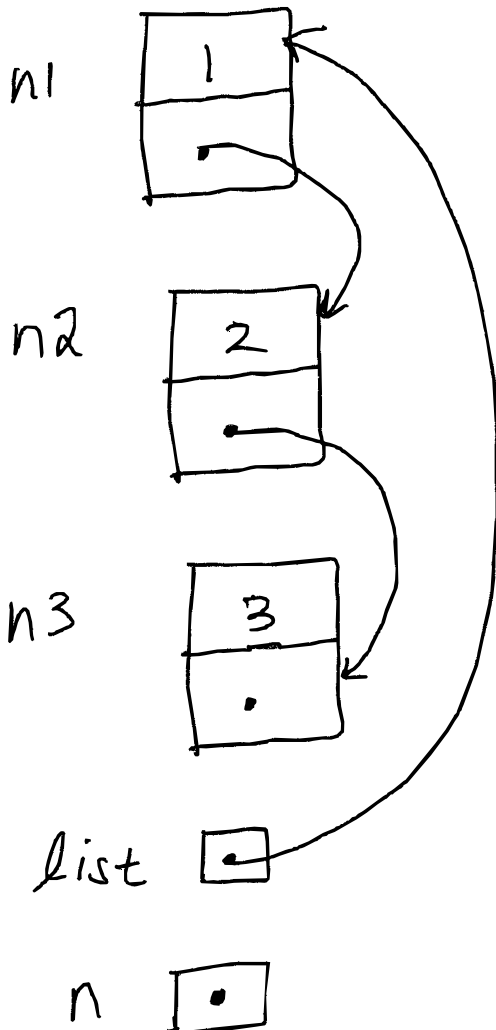
Linked list: list of nodes
connected using next/prev
pointers

USES:

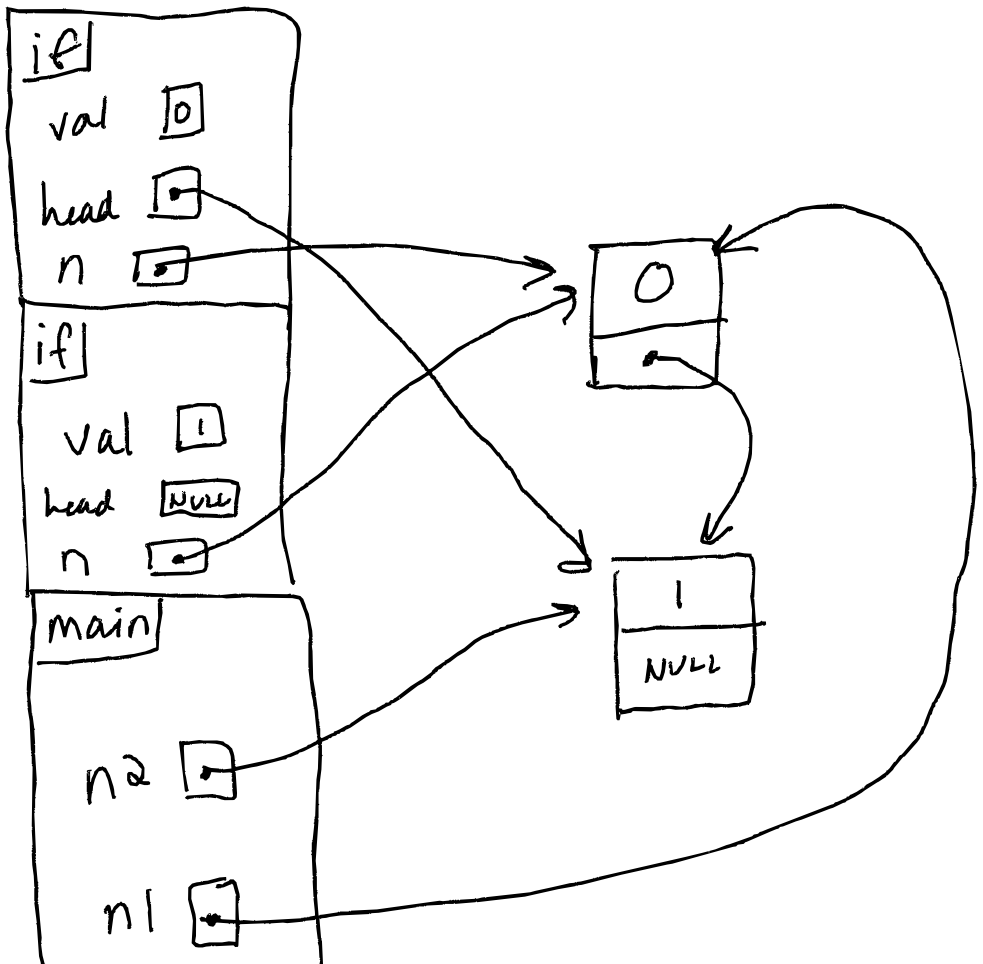
- fast insert (uses less mem than array typically)
- great for stacks

Visualize the LL from

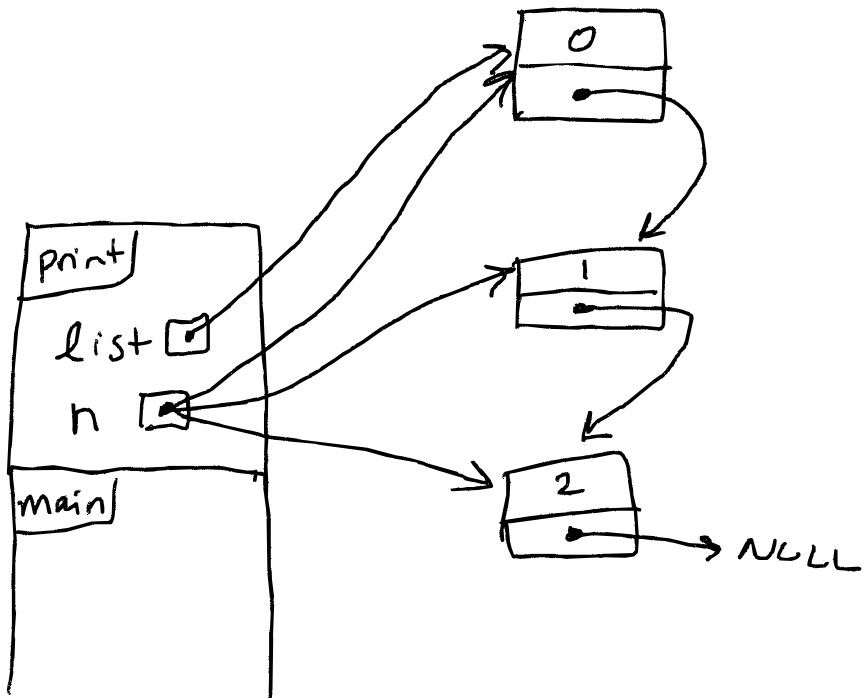
this code



Visualize insert-front



Visualize Print



Exercise : Draw the
stack diagram

* 8 slides to teach myself
for Monday

*