

CS 103000

Prof. Madeline Blount

Week 11:

POINTERS

attendance link:

<https://cs103-week11.glitch.me/>



Dall-E 2: cats learning C++ in the forest on '90's technology

THE MYSTIQUE OF POINTERS

tricky but
IMPORTANT ...



<https://cs103-week11.glitch.me/>

https://www.youtube.com/watch?v=2ybLD6_2gKM

THE MYSTIQUE OF POINTERS

tricky but
IMPORTANT ...



<https://cs103-week11.glitch.me/>

<https://www.youtube.com/watch?v=DTxHyVn0ODg>

THE MYSTIQUE OF POINTERS

tricky but
IMPORTANT ...



<https://cs103-week11.glitch.me/>

<https://www.youtube.com/watch?v=t5NszblerYc>

THE MYSTIQUE OF POINTERS

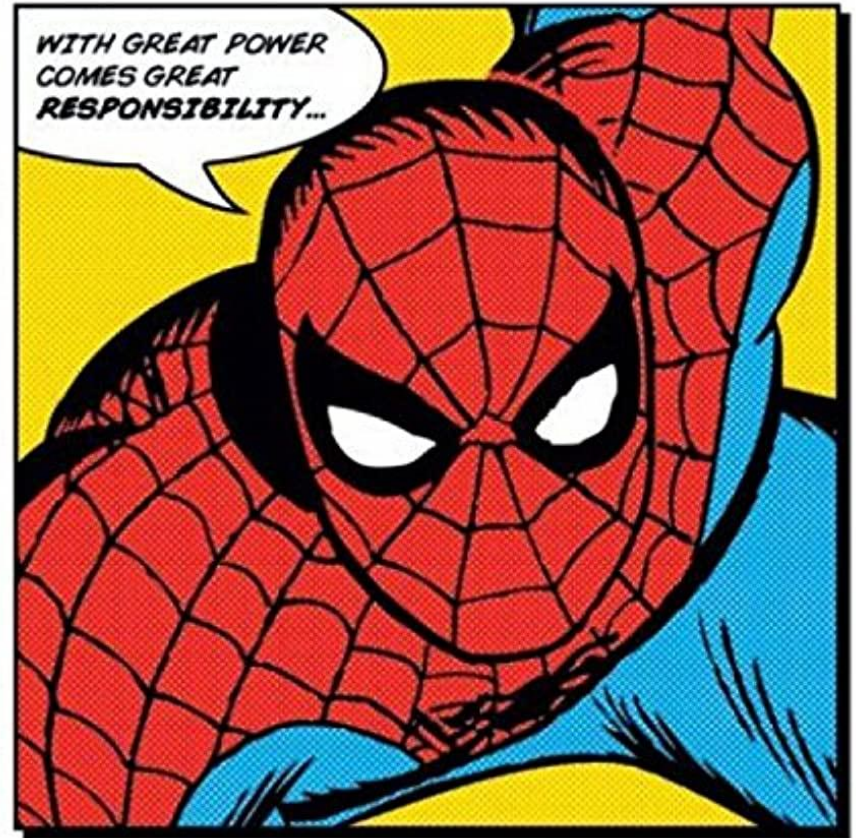
tricky but
IMPORTANT ...



<https://cs103-week11.glitch.me/>

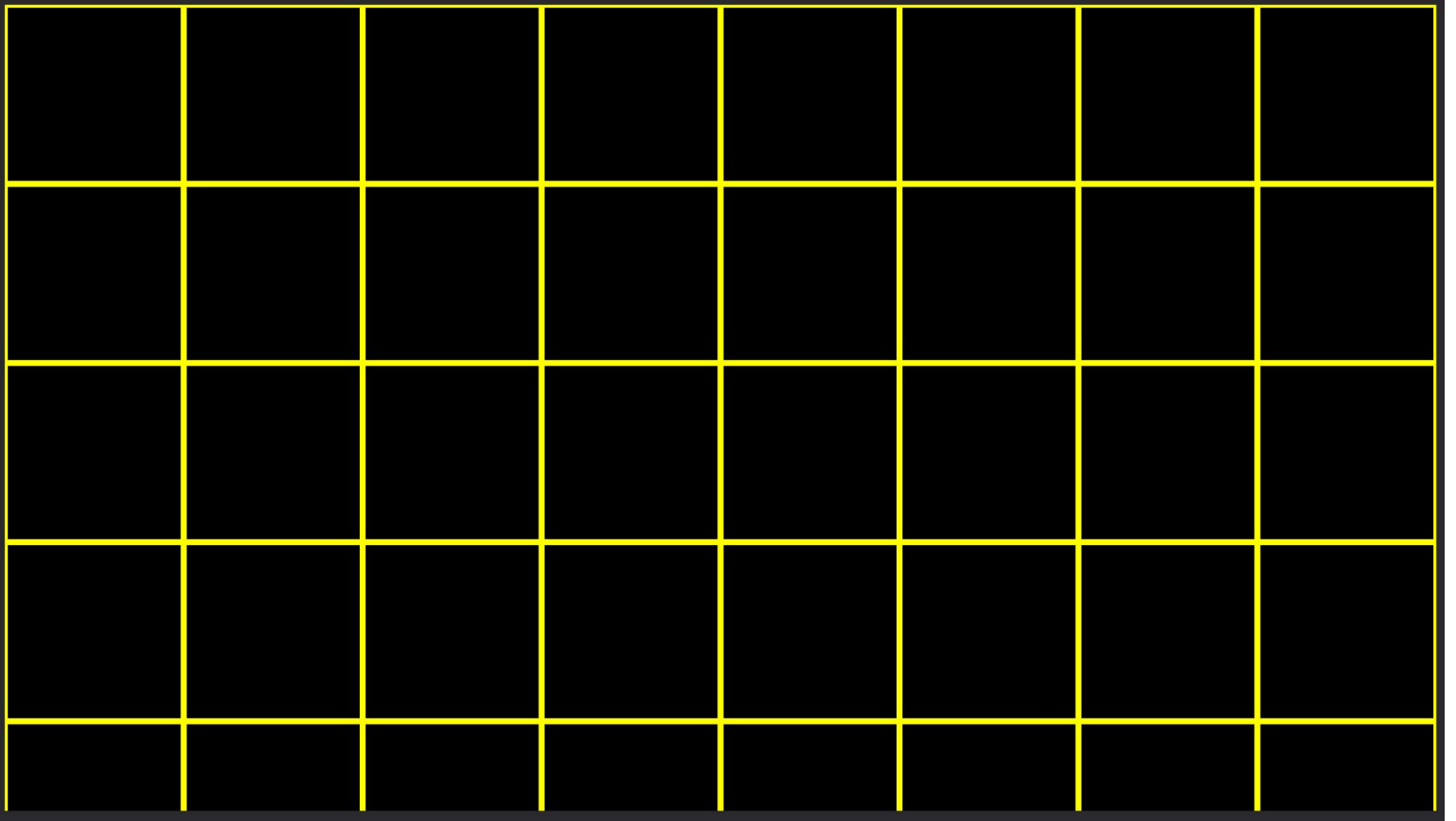
https://www.youtube.com/watch?v=i49_SNt4yfk

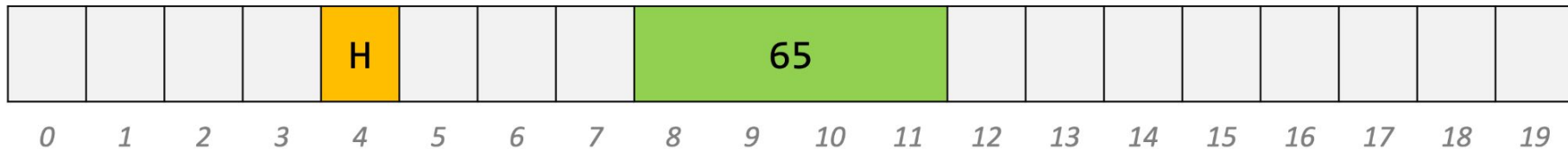
POINTERS give you
the power to manage
and directly
manipulate **MEMORY**



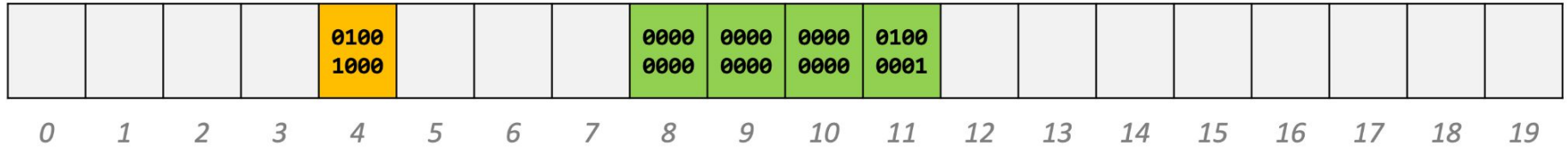
MEMORY: for programs, RAM (random access memory)







```
char c = 'H';  
int speedlimit = 65;
```

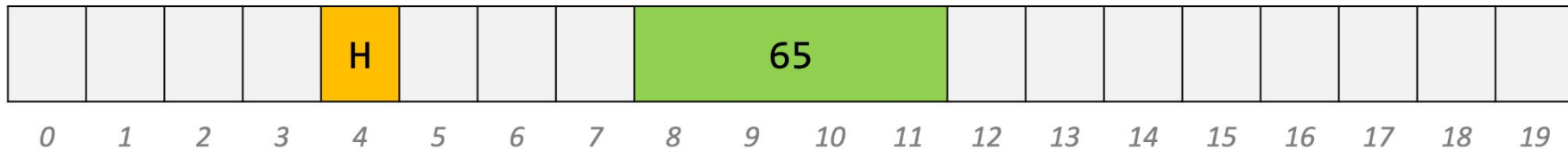


```
char c = 'H';  
int speedlimit = 65;
```

Memory Size

The size of a given data type is measured in bytes:

Data Type	Memory Size
<code>bool</code>	1 byte
<code>char</code>	1 byte
<code>int</code>	4 bytes
<code>float</code>	4 bytes
<code>double</code>	8 bytes
<code>std::string</code>	24 bytes



```
char c = 'H';  
int speedlimit = 65;
```




1

2

3

4



POINTER = MEMORY ADDRESS

POINTER = MEMORY ADDRESS

POINTER = integer that
stores a memory address

POINTER = MEMORY ADDRESS

POINTER = variable that
holds an integer which is
a memory address

POINTER = MEMORY ADDRESS

0x16d9d7470

hexadecimal

0 1 2 3 4 5 6 7 8 9 a b c d e f

16 1

#

16 1

02

16 1

09

16 1

θA

11111111

FF

POINTER = MEMORY ADDRESS

0x16d9d7470

6134002800

101101101100111010111010001110000

MAN, I SUCK AT THIS GAME.
CAN YOU GIVE ME
A FEW POINTERS?

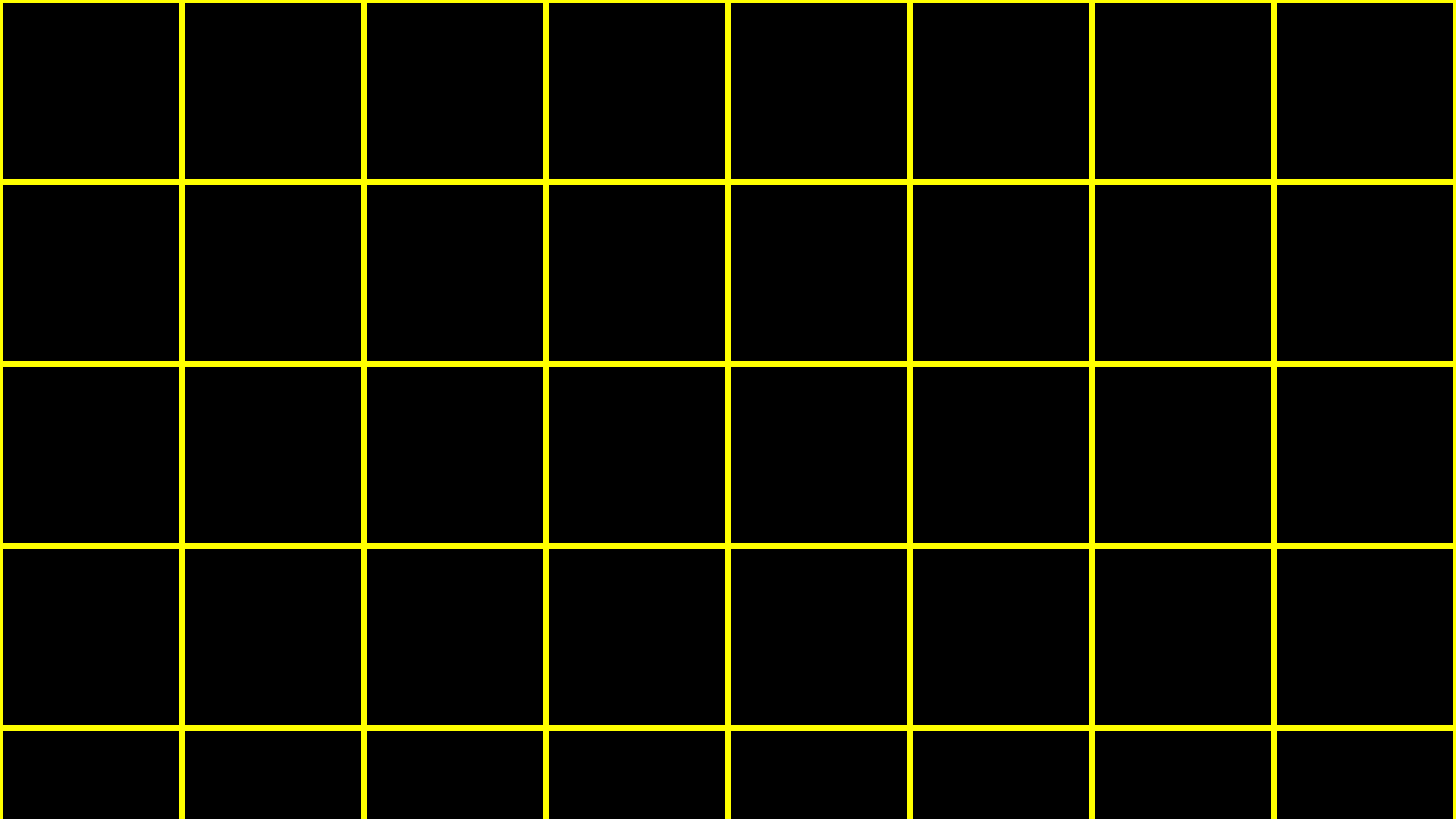
|
I HATE YOU.

0x3A28213A
0x6339392C,
0x7363682E.



0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7
0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17
0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F
from Harvard CS50							

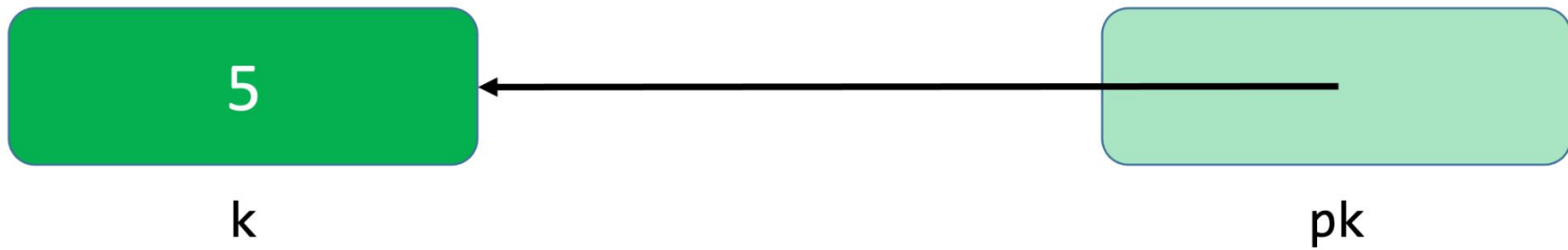

```
int n = 50;
```

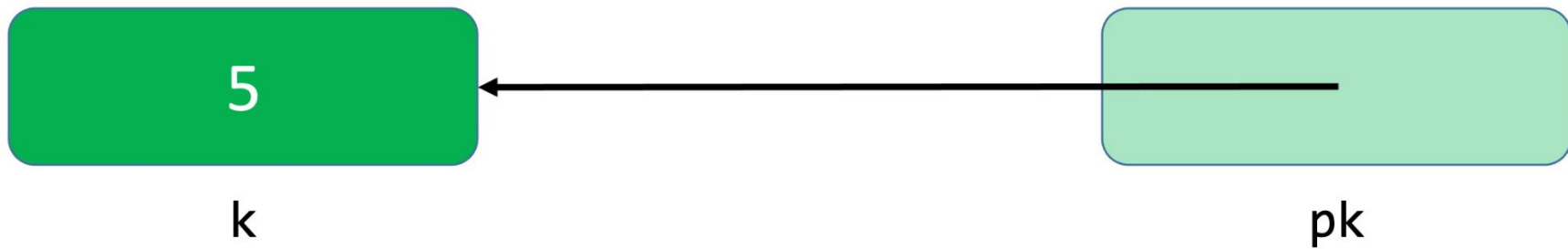


50
n

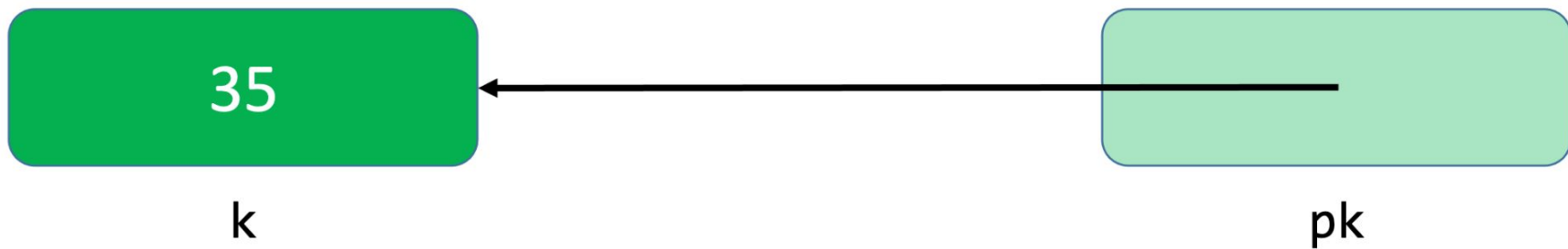
50

0x123

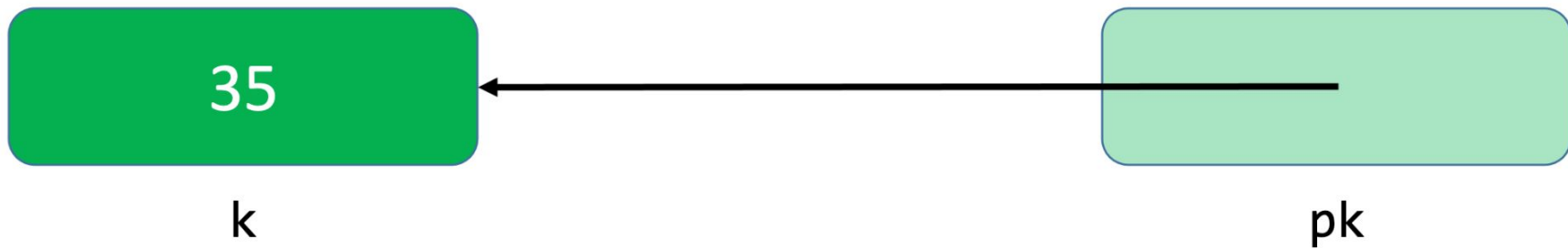




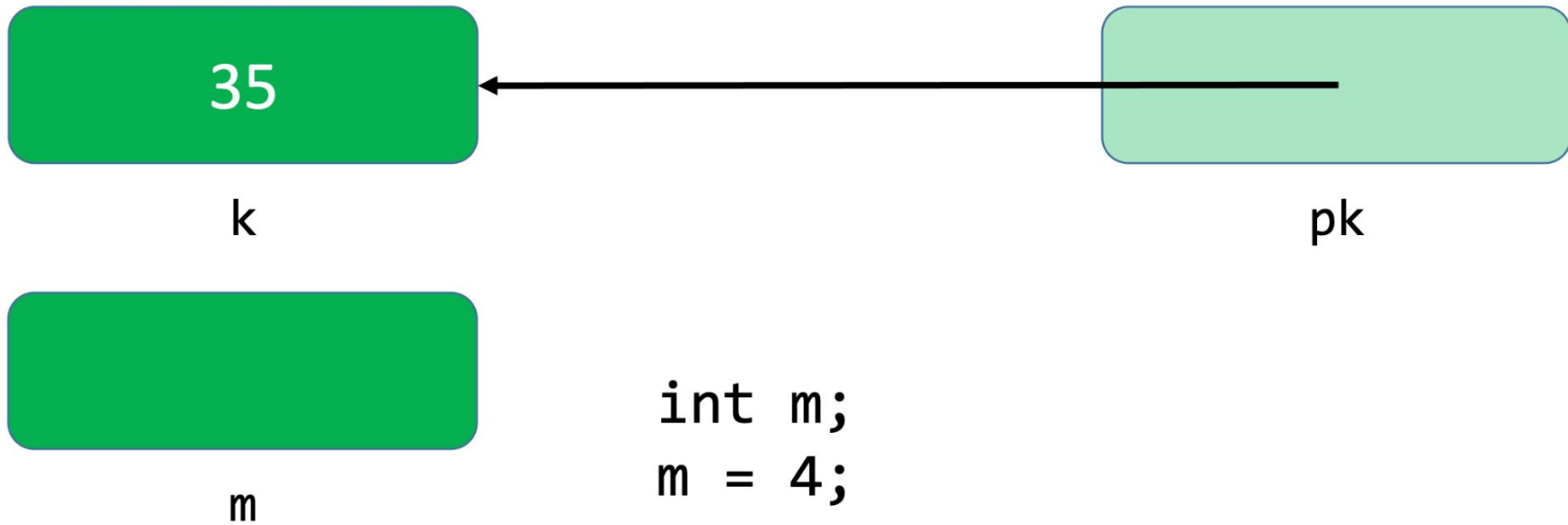
`*pk = 35;`

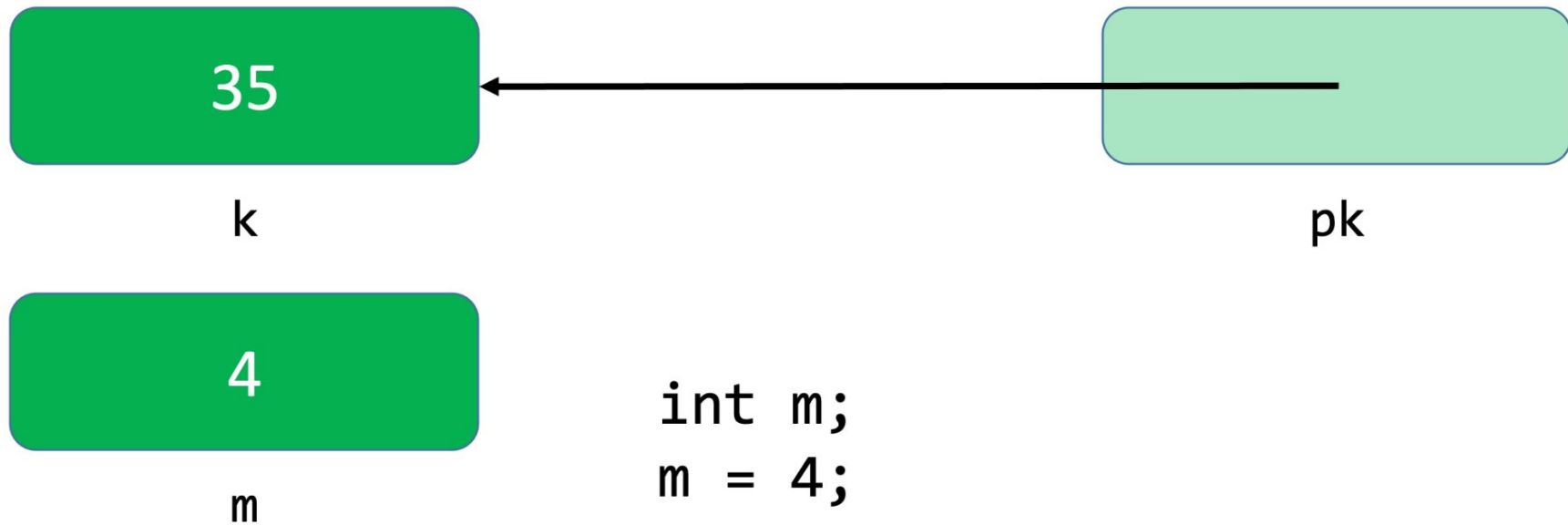


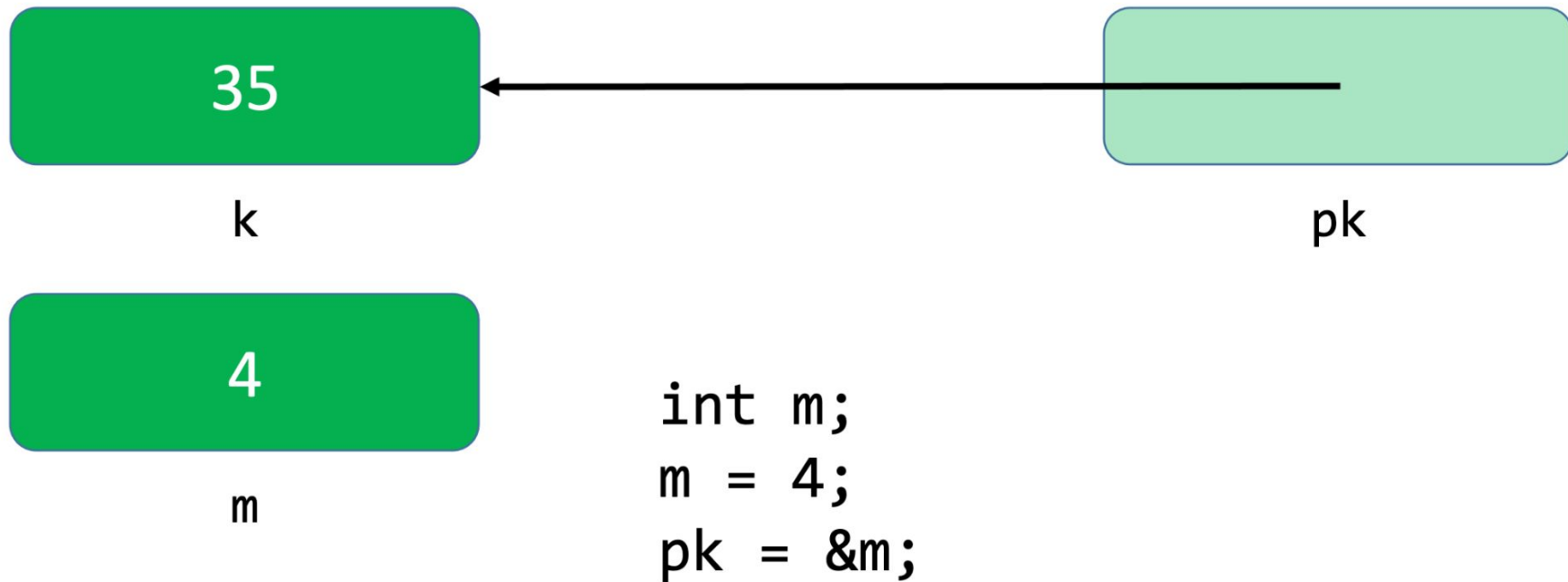
`*pk = 35;`



```
int m;
```



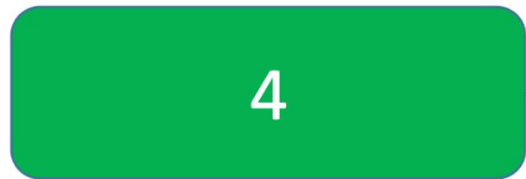




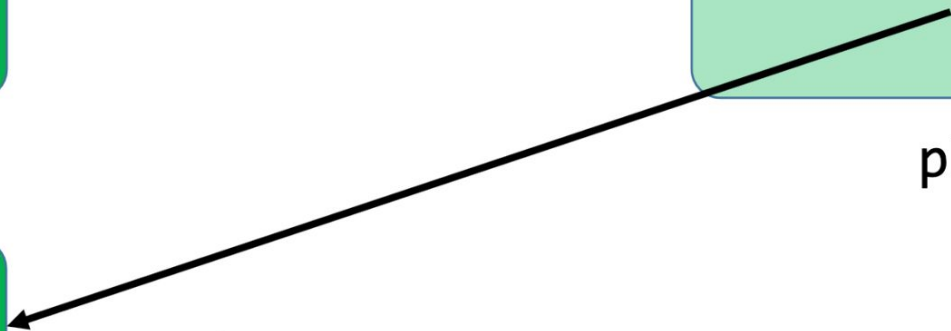
`k`



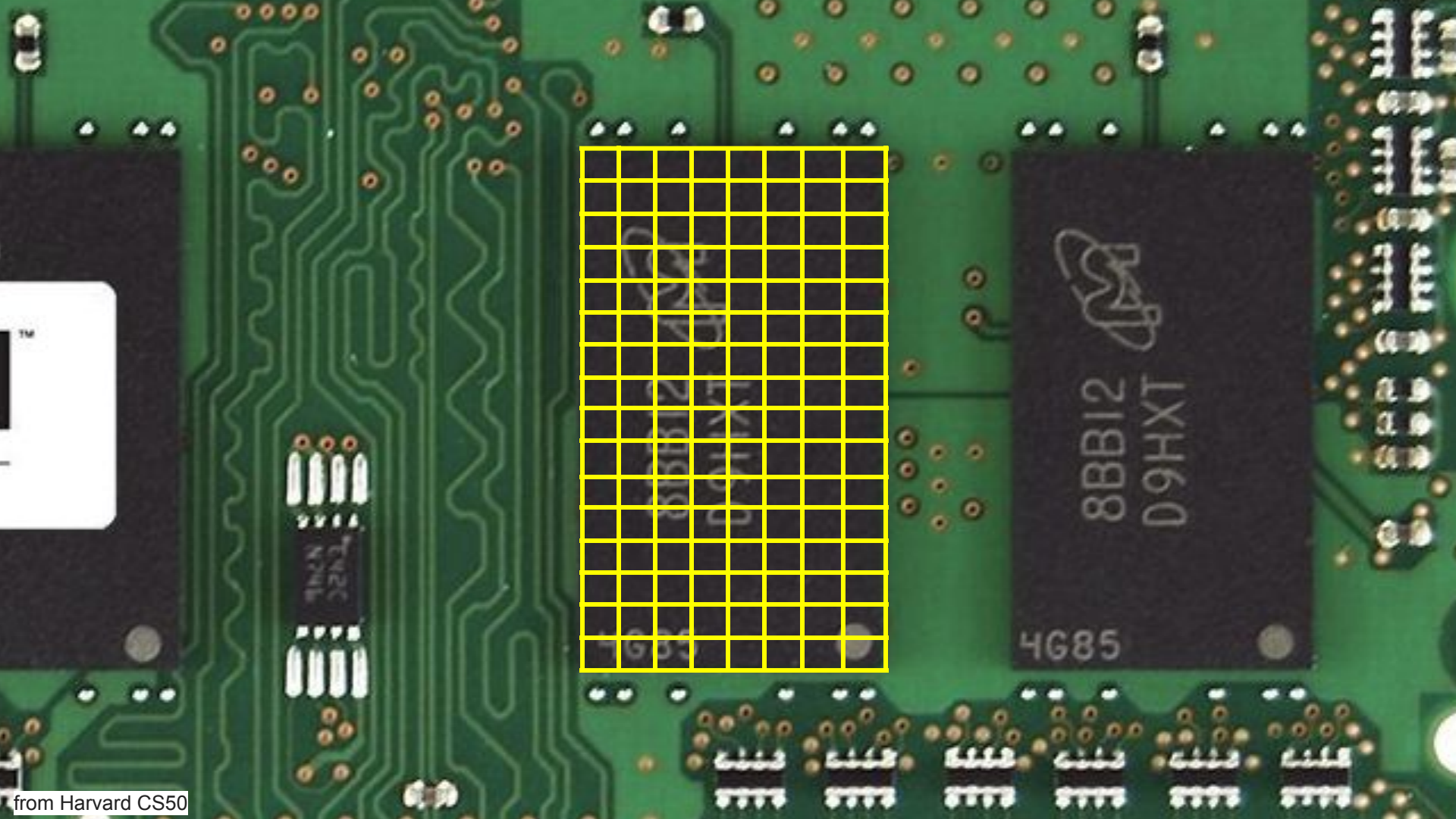
`pk`

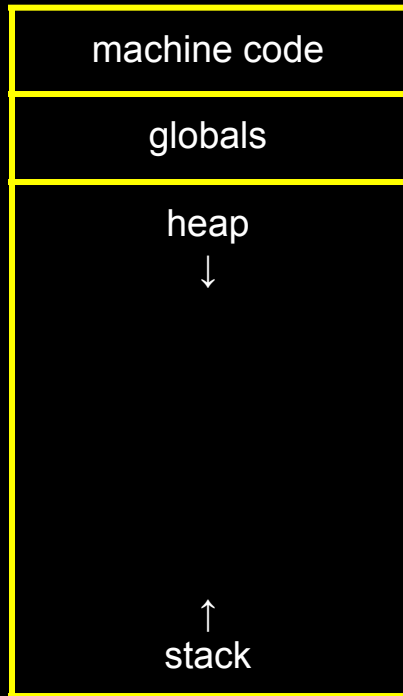


`m`



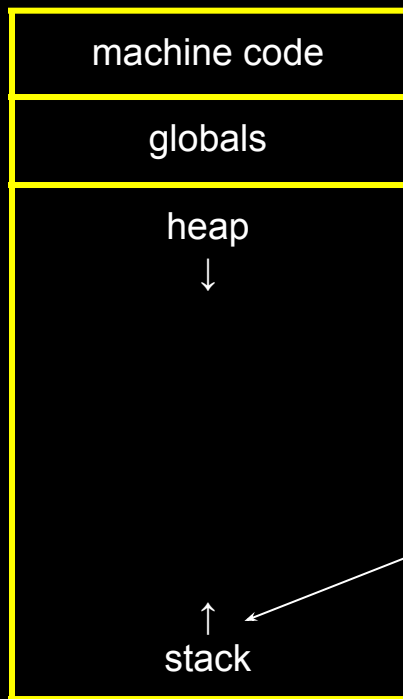
```
int m;  
m = 4;  
pk = &m;
```





new keyword

control: we get to
allocate (and must
free with **delete**)



























local scope

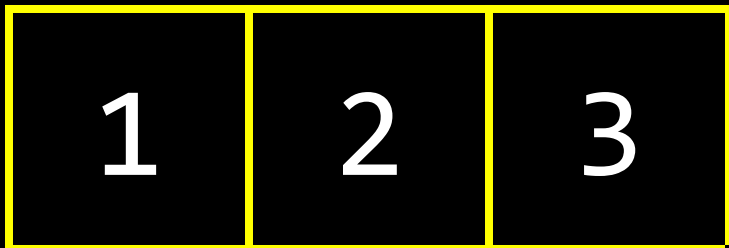
control: clears by
itself, after every
function

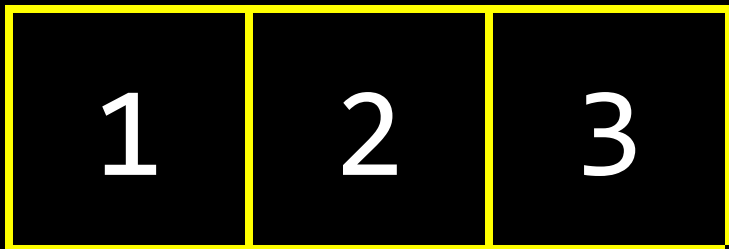


```
x = new int;
```

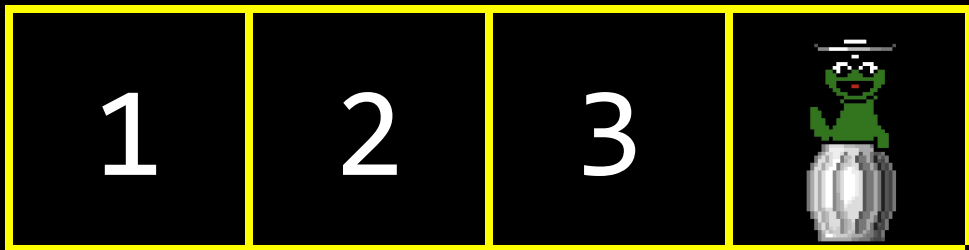
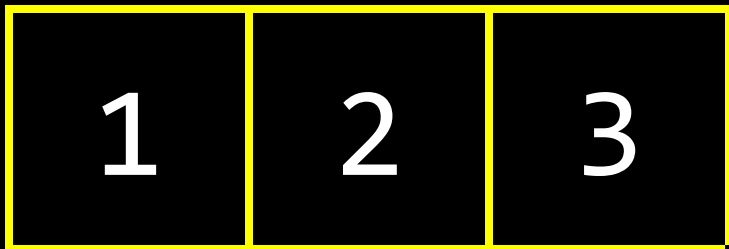

	1	2	3				

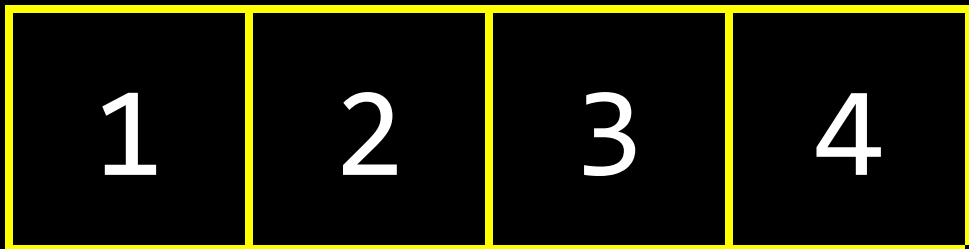
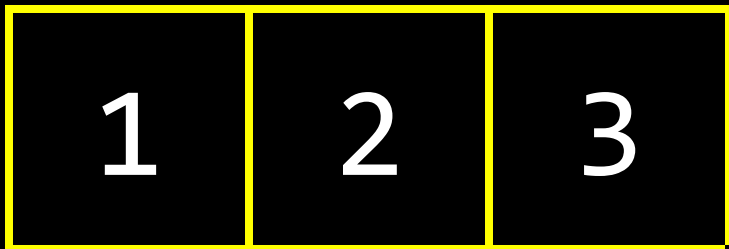
							
	1	2	3	h	e	l	l
o	,		w	o	r	l	d
\0							
							

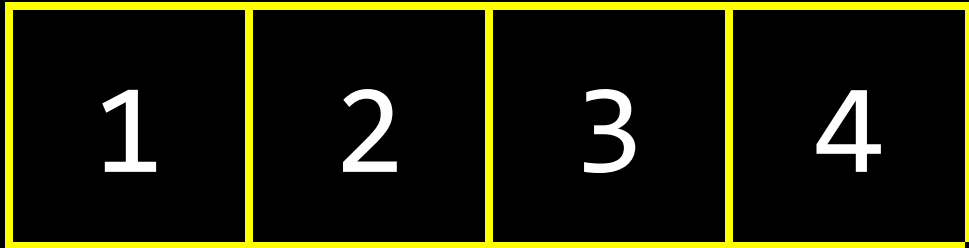




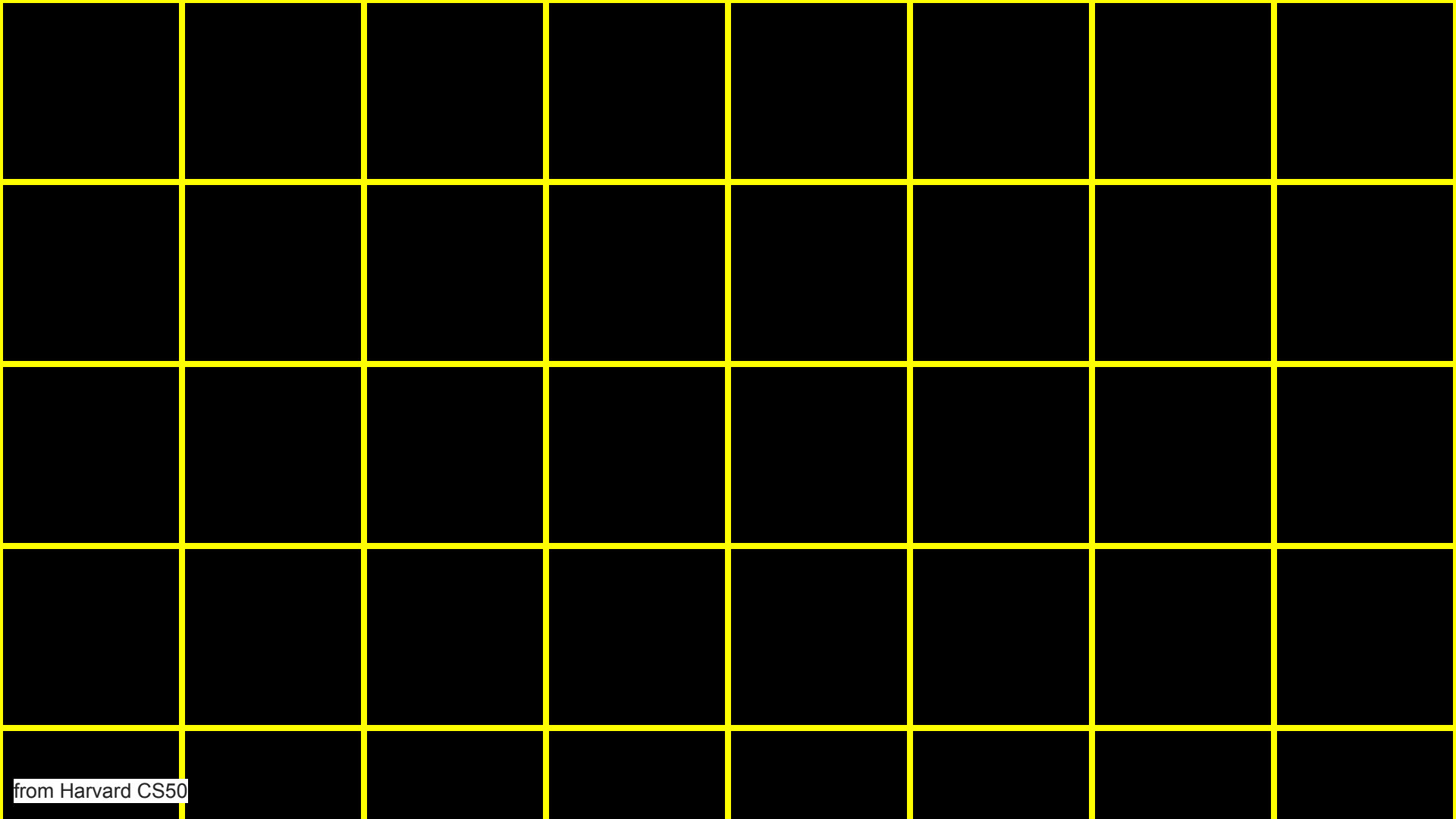








linked lists



1

0x123

1

0x123

2

0x456

1

0x123

2

0x456

3

0x789

1

0x123

2

0x456

3

0x789

1

0x123

0x456

2

0x456

3

0x789

1

0x123

0x456

2

0x456

0x789

3

0x789

1

0x123

0x456

2

0x456

0x789

3

0x789

0x0

1

0x123

0x456

2

0x456

0x789

3

0x789

NULL

