

This is CS50



CS50
STRESS BALL

CS50
STRESS BALL

CS
STRESS

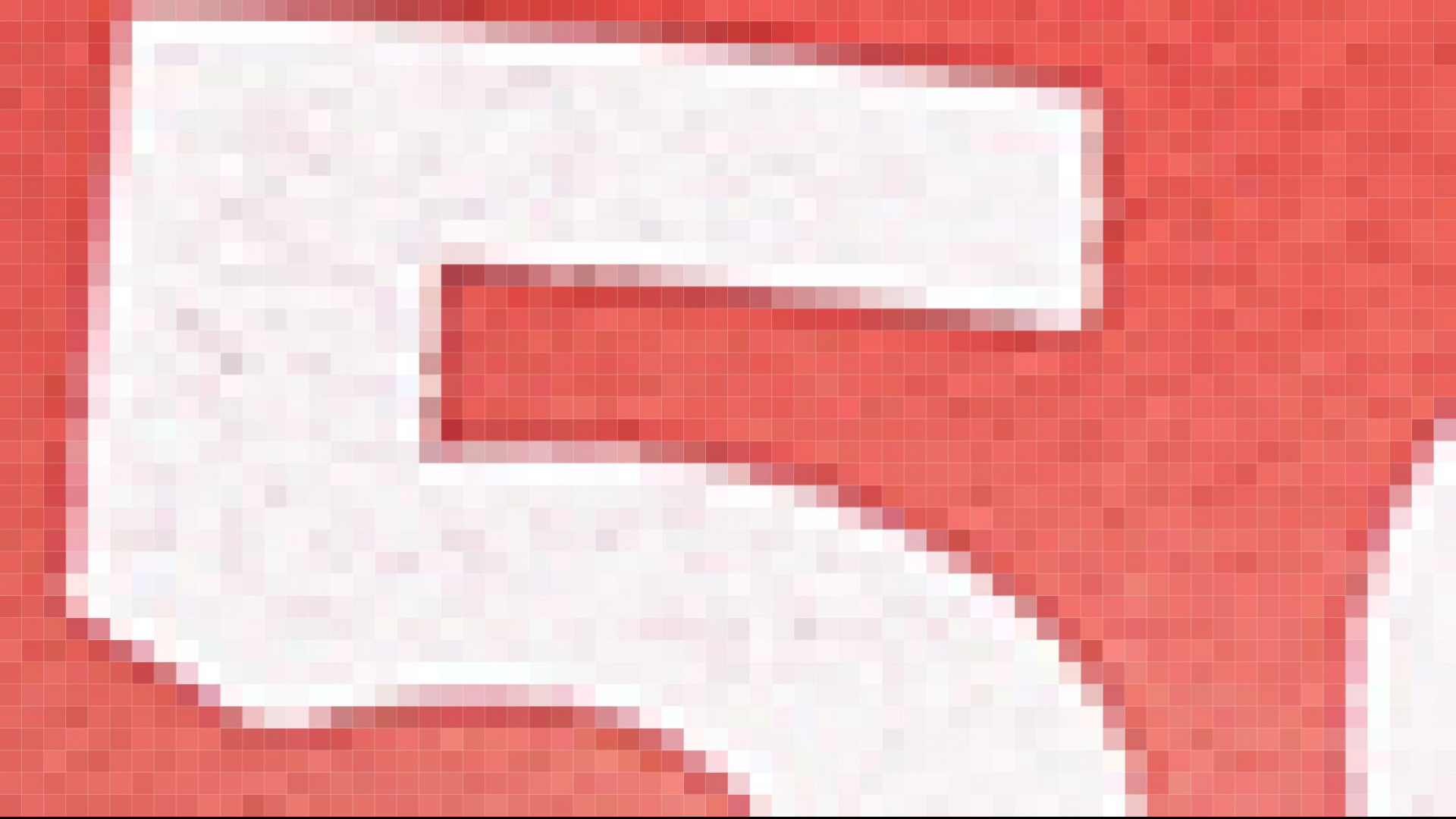


CS50
STRESS BALL

CS
STRESS

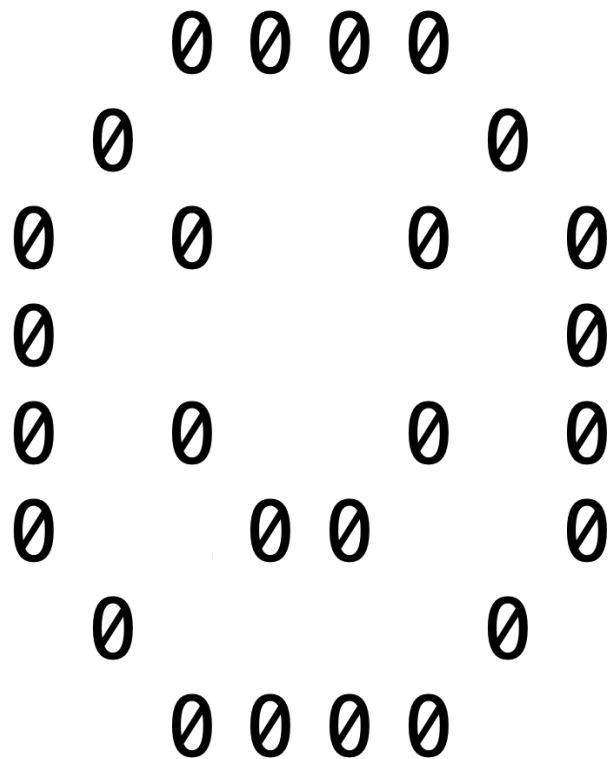
CS50
STRESS BALL

50

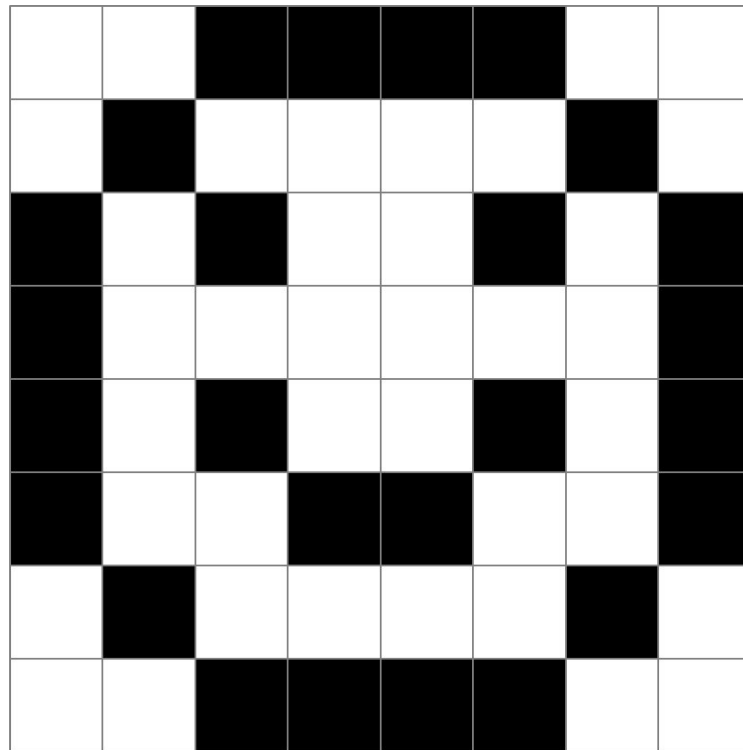




1	1	0	0	0	0	1	1
1	0	1	1	1	1	0	1
0	1	0	1	1	0	1	0
0	1	1	1	1	1	1	0
0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	0
1	0	1	1	1	1	0	1
1	1	0	0	0	0	1	1



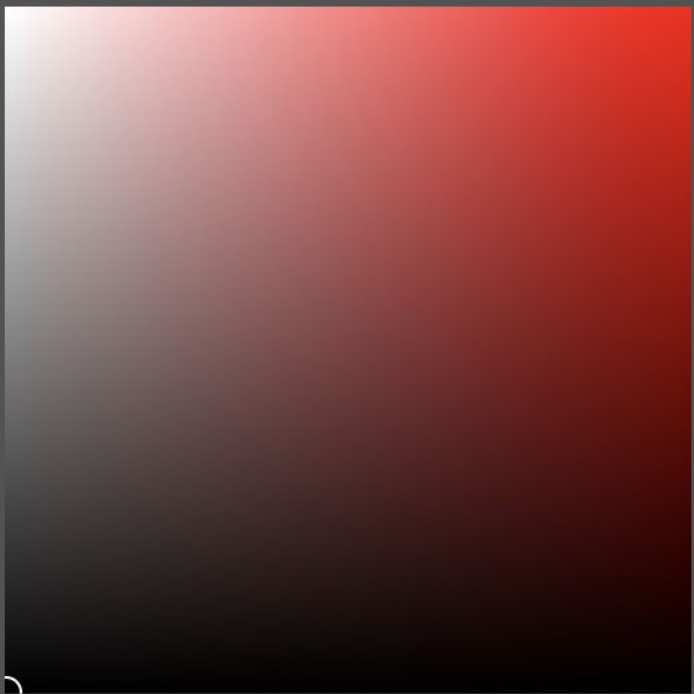
1	1	0	0	0	0	1	1
1	0	1	1	1	1	0	1
0	1	0	1	1	0	1	0
0	1	1	1	1	1	1	0
0	1	0	1	1	0	1	0
0	1	1	0	0	1	1	0
1	0	1	1	1	1	0	1
1	1	0	0	0	0	1	1



cs50.ly/art

RGB

Color Picker (Foreground Color)



☐ Only Web Colors

new



current

☒ H: 0 °

☐ S: 0 %

☐ B: 0 %

☐ R: 0

☐ G: 0

☐ B: 0

☐ L: 0

☐ a: 0

☐ b: 0

C: 75 %

M: 68 %

Y: 67 %

K: 90 %

000000

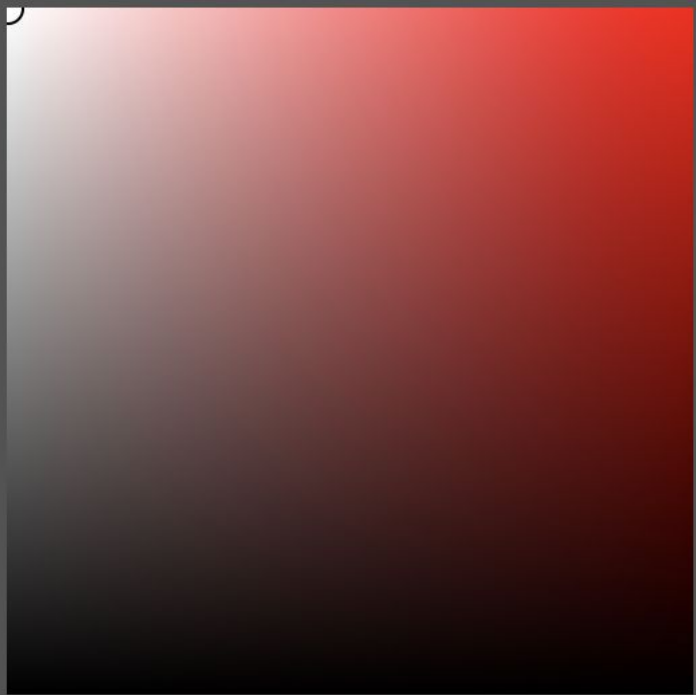
OK

Cancel

Add to Swatches

Color Libraries

Color Picker (Foreground Color)



new



current

OK

Cancel

Add to Swatches

Color Libraries

☒ H: 0 °

☐ L: 100

☐ S: 0 %

☐ a: 0

☐ B: 100 %

☐ b: 0

☐ R: 255

C: 0 %

☐ G: 255

M: 0 %

☐ B: 255

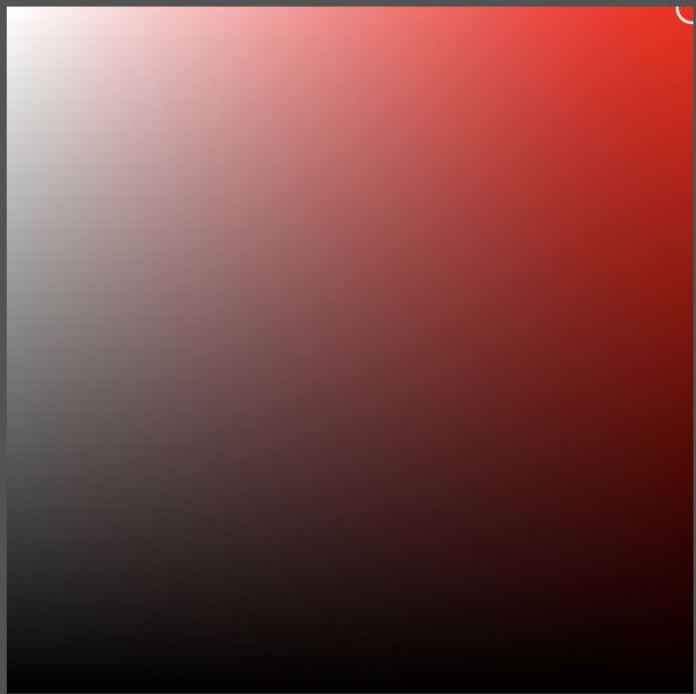
Y: 0 %

K: 0 %

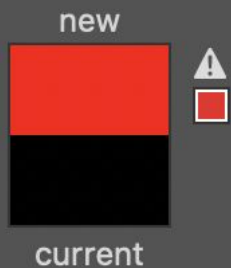
☐ Only Web Colors

FFFFFFFF

Color Picker (Foreground Color)



☐ Only Web Colors



OK

Cancel

Add to Swatches

Color Libraries

☒ H: 0 °

☐ L: 54

☐ S: 100 %

☐ a: 81

☐ B: 100 %

☐ b: 70

☐ R: 255

C: 0 %

☐ G: 0

M: 99 %

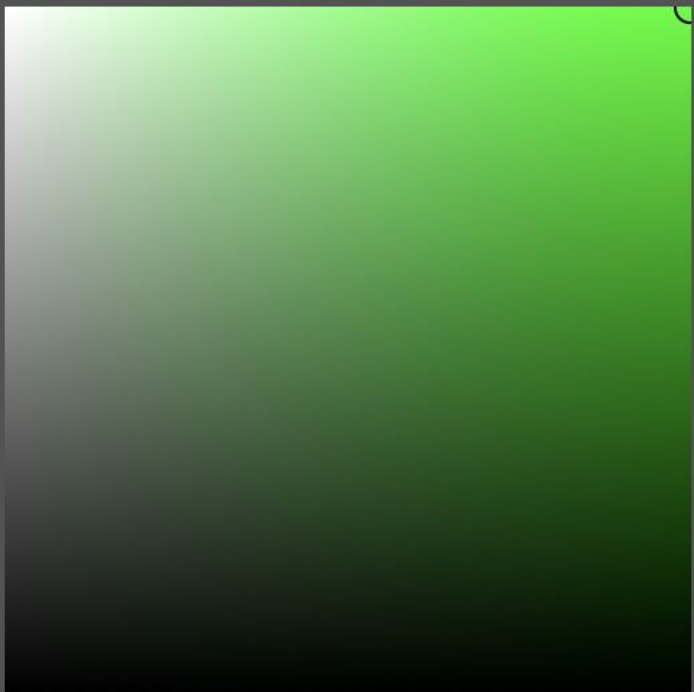
☐ B: 0

Y: 100 %

K: 0 %

FF0000

Color Picker (Foreground Color)



☐ Only Web Colors



OK

Cancel

Add to Swatches

Color Libraries

☒ H: 120 °

☐ L: 88

☐ S: 100 %

☐ a: -79

☐ B: 100 %

☐ b: 81

☐ R: 0

C: 63 %

☐ G: 255

M: 0 %

☐ B: 0

Y: 100 %

00FF00

K: 0 %

Color Picker (Foreground Color)



☐ Only Web Colors

new



current

OK

Cancel

Add to Swatches

Color Libraries

☒ H: 240 °

☐ L: 30

☐ S: 100 %

☐ a: 68

☐ B: 100 %

☐ b: -112

☐ R: 0

C: 88 %

☐ G: 0

M: 77 %

☐ B: 255

Y: 0 %

K: 0 %

0000FF

0 1

0 1 2 3 4 5 6 7 8 9

0 1 2 3 4 5 6 7 8 9 A B C D E F

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

hexadecimal

base-16

16^1 16^0

#

16 1

#

16 1

00

16 1

01

16 1

02

16 1

03

16 1

04

16 1

05

16 1

06

16 1

07

16 1

08

16 1

09

16 1

0A

16 1

ØB

16 1

0C

16 1

ØD

16 1

0E

16 1

ØF

16 1

10

16 1

11

16 1

12

16 1

13

16 1

14

16 1

16 1

FF

16 1

FF

$16 \times F + 1 \times F$

16 1

FF

$16 \times 15 + 1 \times 15$

16 1

FF

240 + 15

16 1

FF

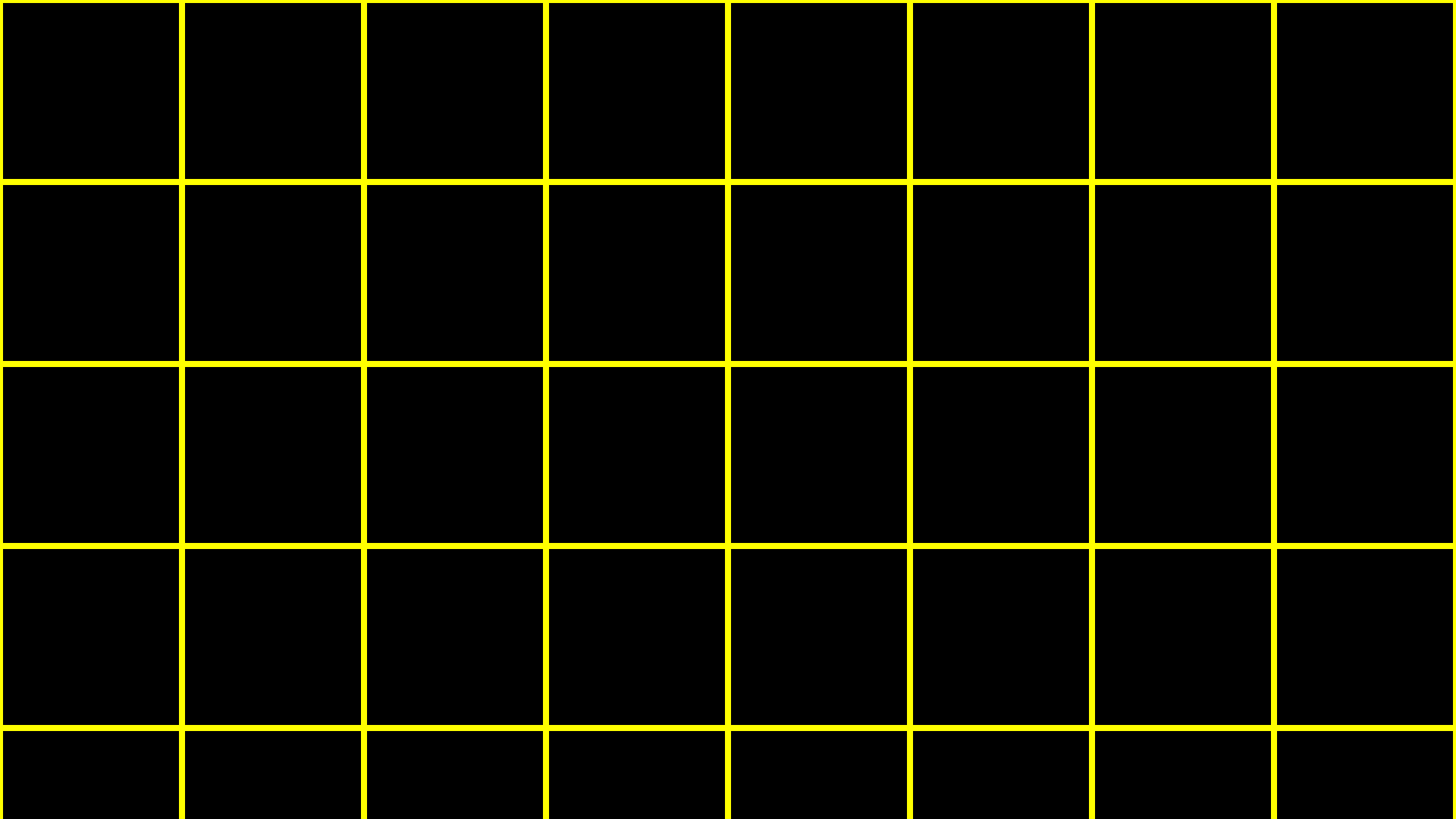
255

F

1111

11111111

FF



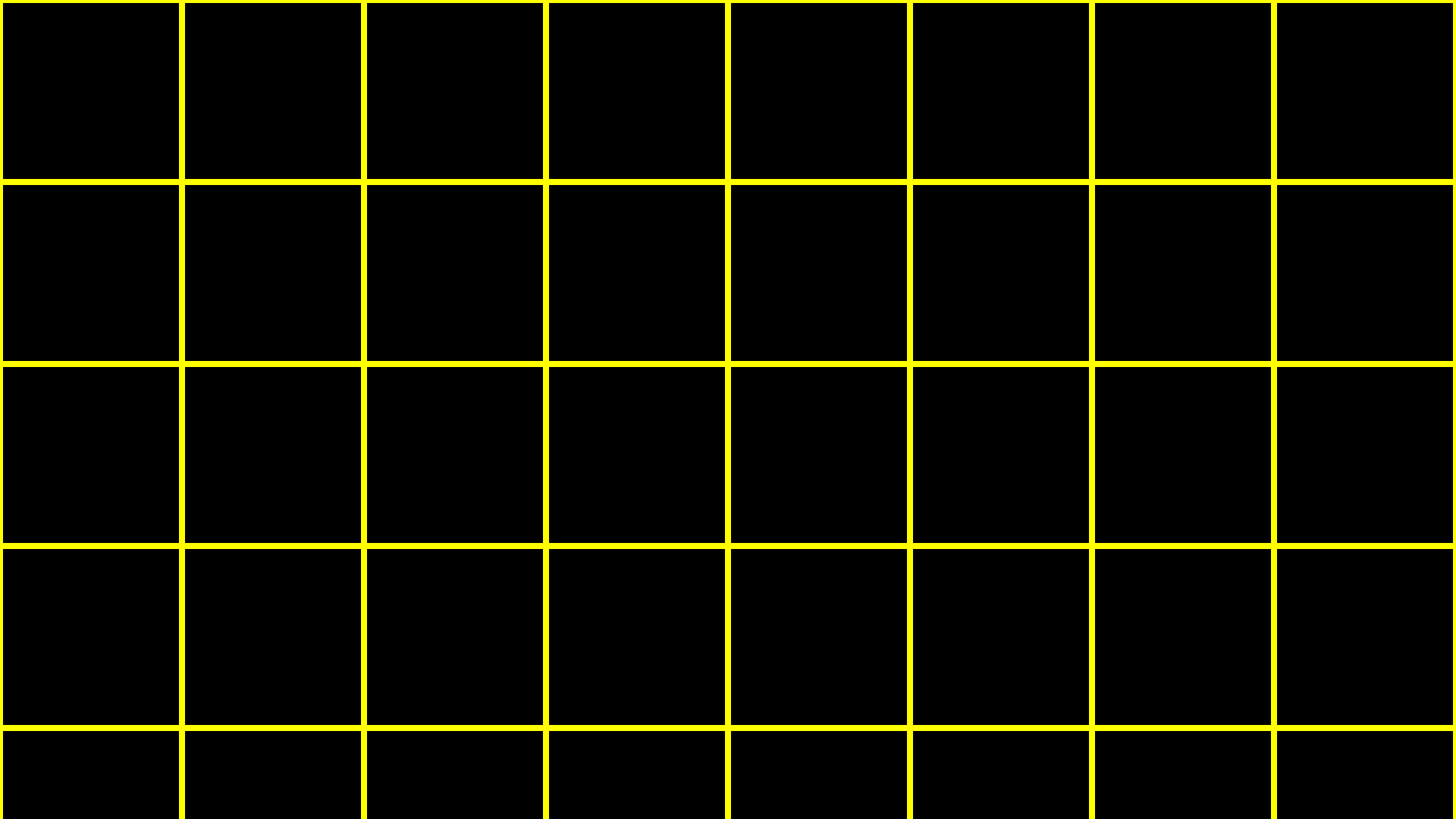
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15

0	1	2	3	4	5	6	7
8	9	A	B	C	D	E	F

0	1	2	3	4	5	6	7
8	9	A	B	C	D	E	F
10	11	12	13	14	15	16	17
18	19	1A	1B	1C	1D	1E	1F

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

```
int n = 50;
```



				50			
						n	

				50			
				0x123			

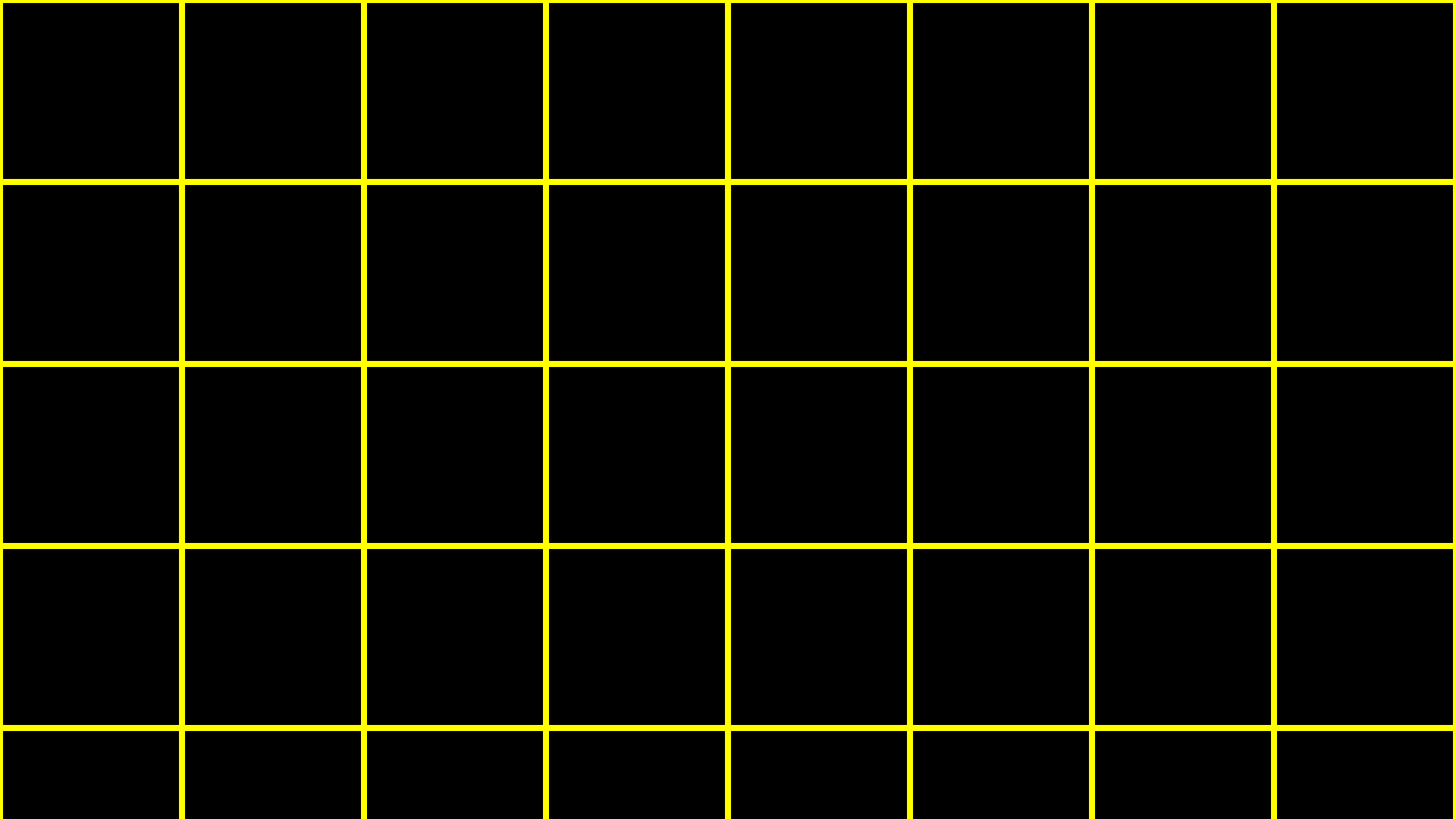
&

*

pointers

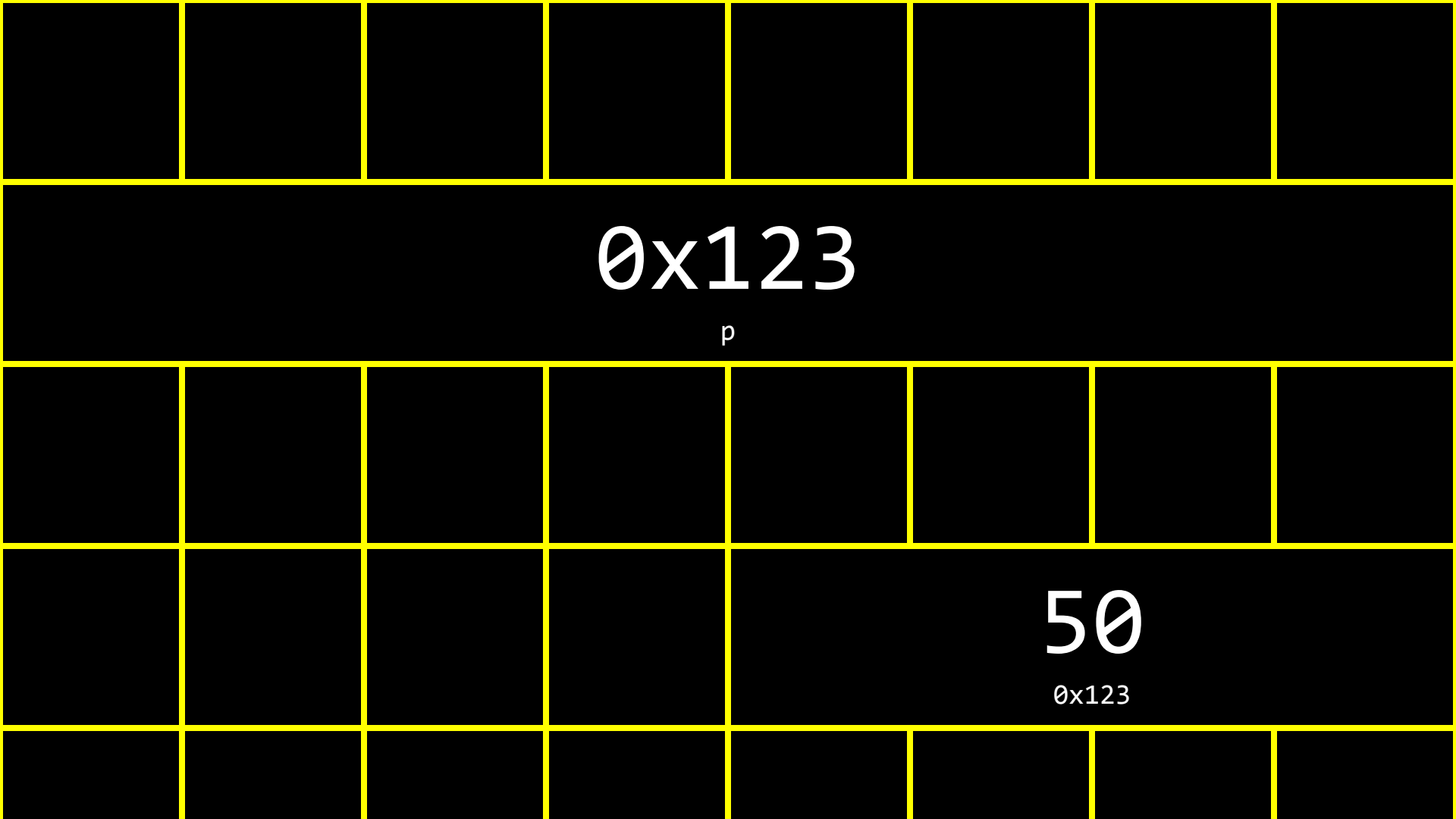
```
int n = 50;
```

```
int *p = &n;
```



				50			
						n	

				50			
				0x123			



0x123

p

50

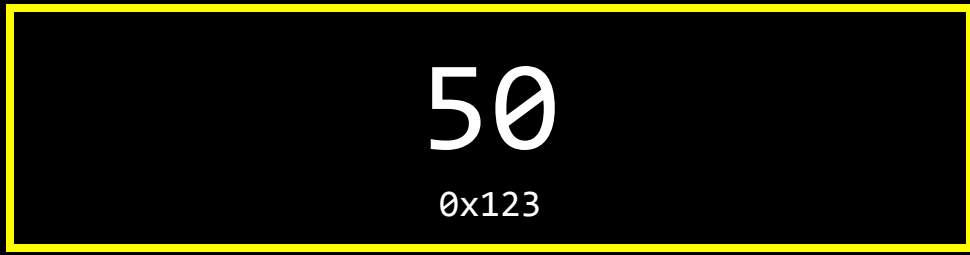
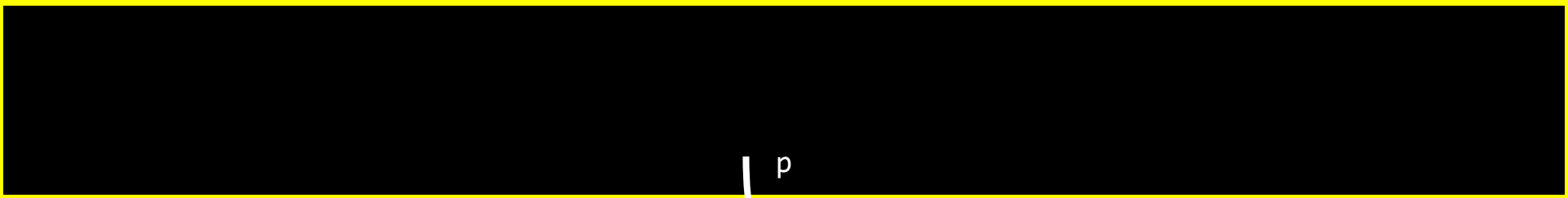
0x123

0x123

p

50

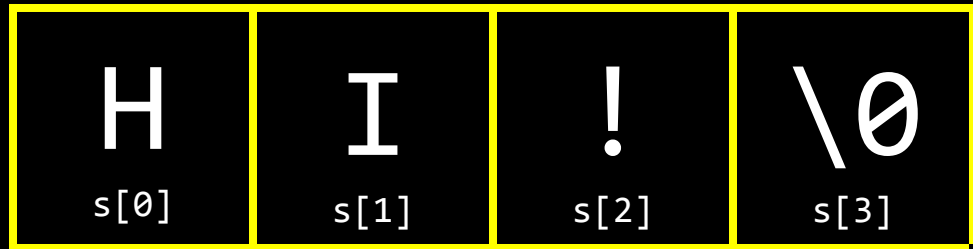
0x123



string

```
string s = "HI!";
```

H	I	!	\0
---	---	---	----



H	I	!	\0
0x123	0x124	0x125	0x126

0x123

s

H

0x123

I

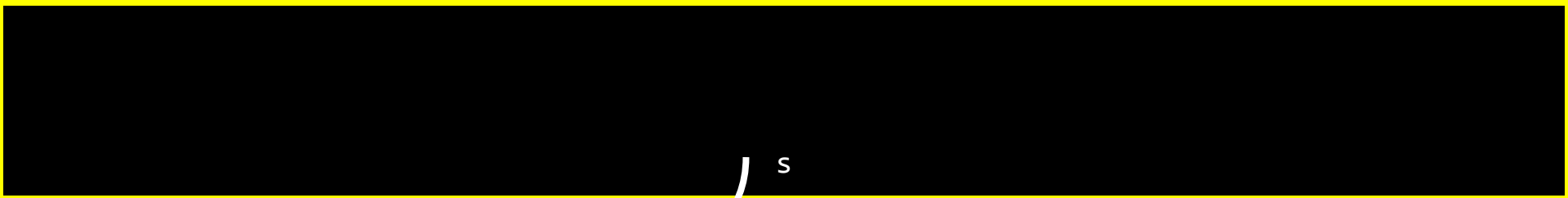
0x124

!

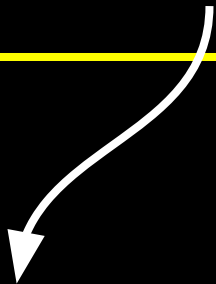
0x125

\0

0x126



s



H	I	!	\0
---	---	---	----

```
string s = "HI!";
```

```
string s = "HI!";
```

```
char *s = "HI!";
```

```
typedef struct
{
    string name;
    string number;
}
person;
```

```
typedef struct
{
    string name;
    string number;
}
person;
```

```
typedef struct
{
    string name;
    string number;
}
person;
```



```
typedef struct
{
    string name;
    string number;
}
person;
```

```
typedef int integer;
```

```
typedef int integer;
```

```
typedef int integer;
```

```
typedef char *string;
```

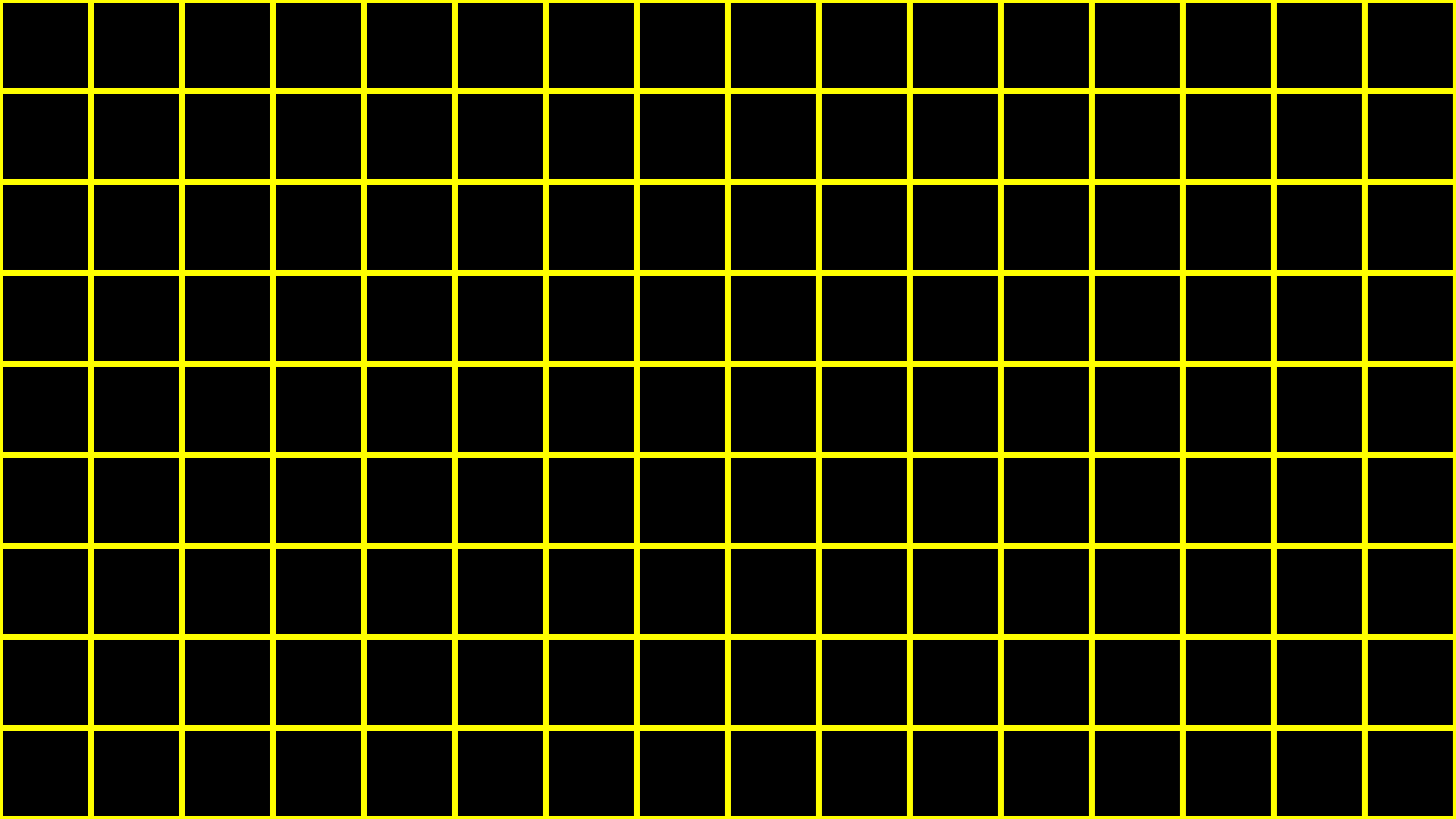
```
typedef char *string;
```

```
typedef char *string;
```

pointer arithmetic



pointer arithmetic



[illegible]

S

I	!
---	---

!	\0
---	----

$\setminus \theta$		
--------------------	--	--

S

H

0x123

I

0x124

!

0x125

 $\backslash \theta$

0x126

0x123
s

0x123
s

H 0x123	I 0x124
------------	------------

H 0x123	I 0x124
------------	------------

!

0x125

!

0x125

$\backslash \emptyset$ 0x126	
---------------------------------	--

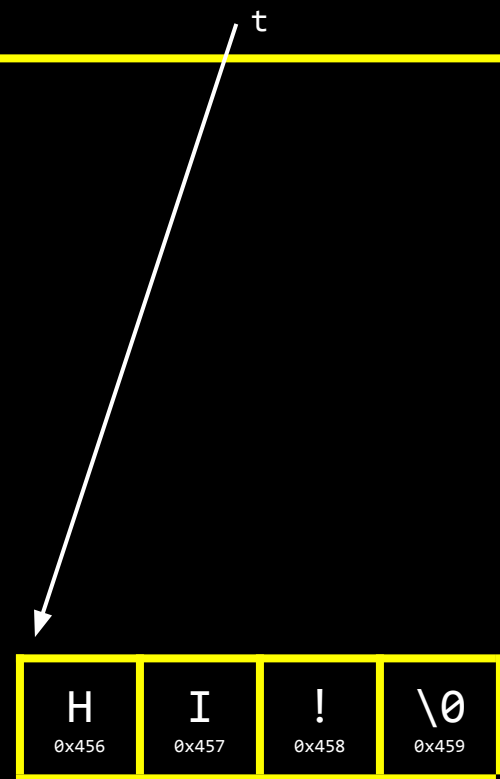
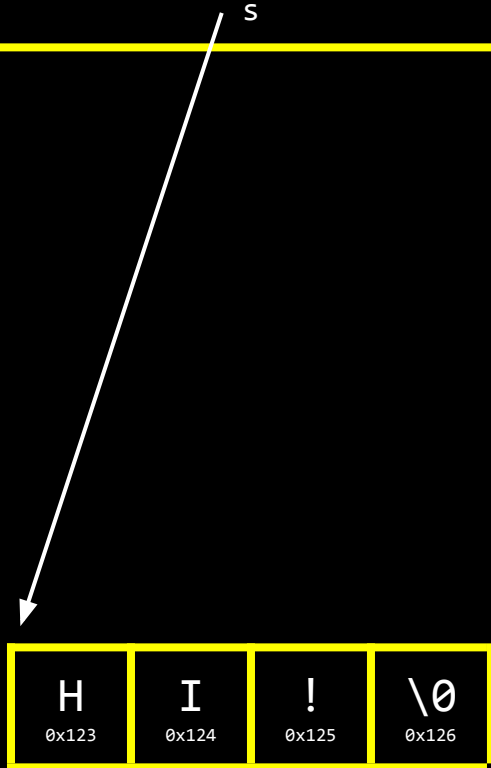
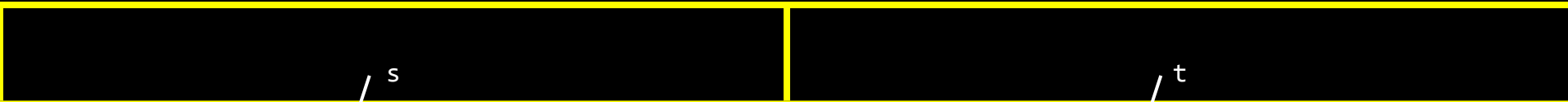
$\backslash \emptyset$ 0x126	
---------------------------------	--

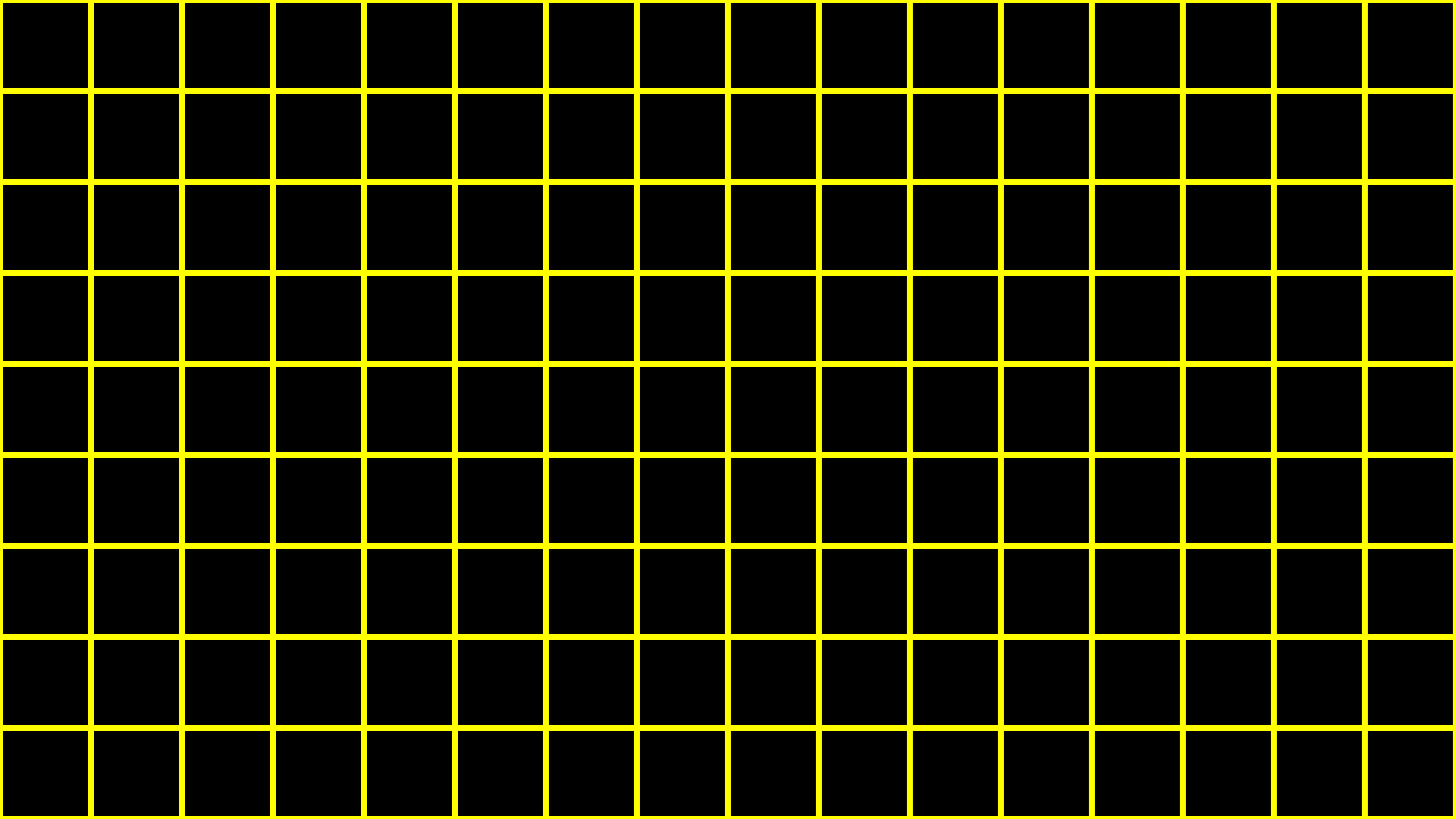
[illegible]

[illegible]

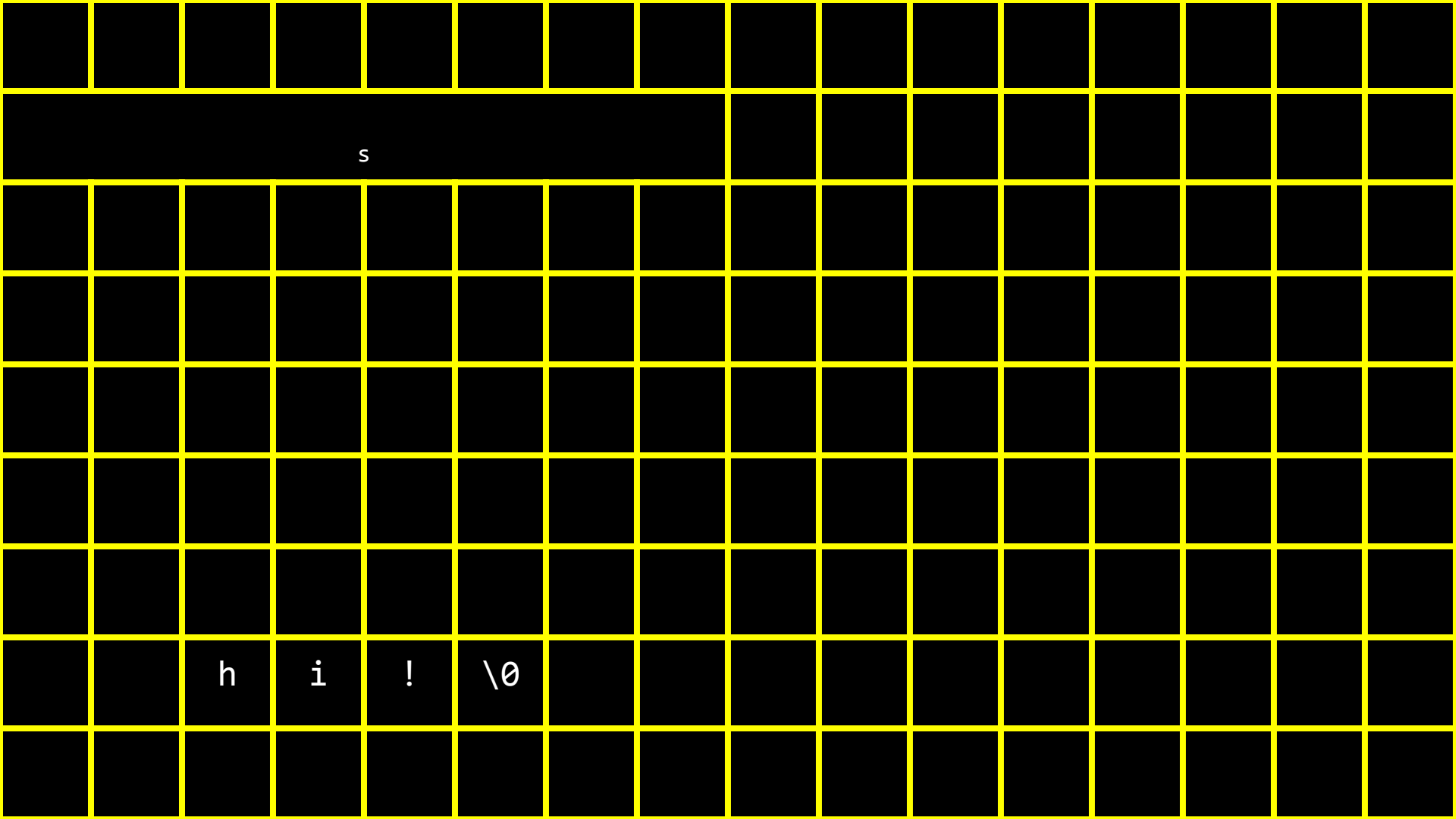
[illegible]

[illegible]





[illegible]



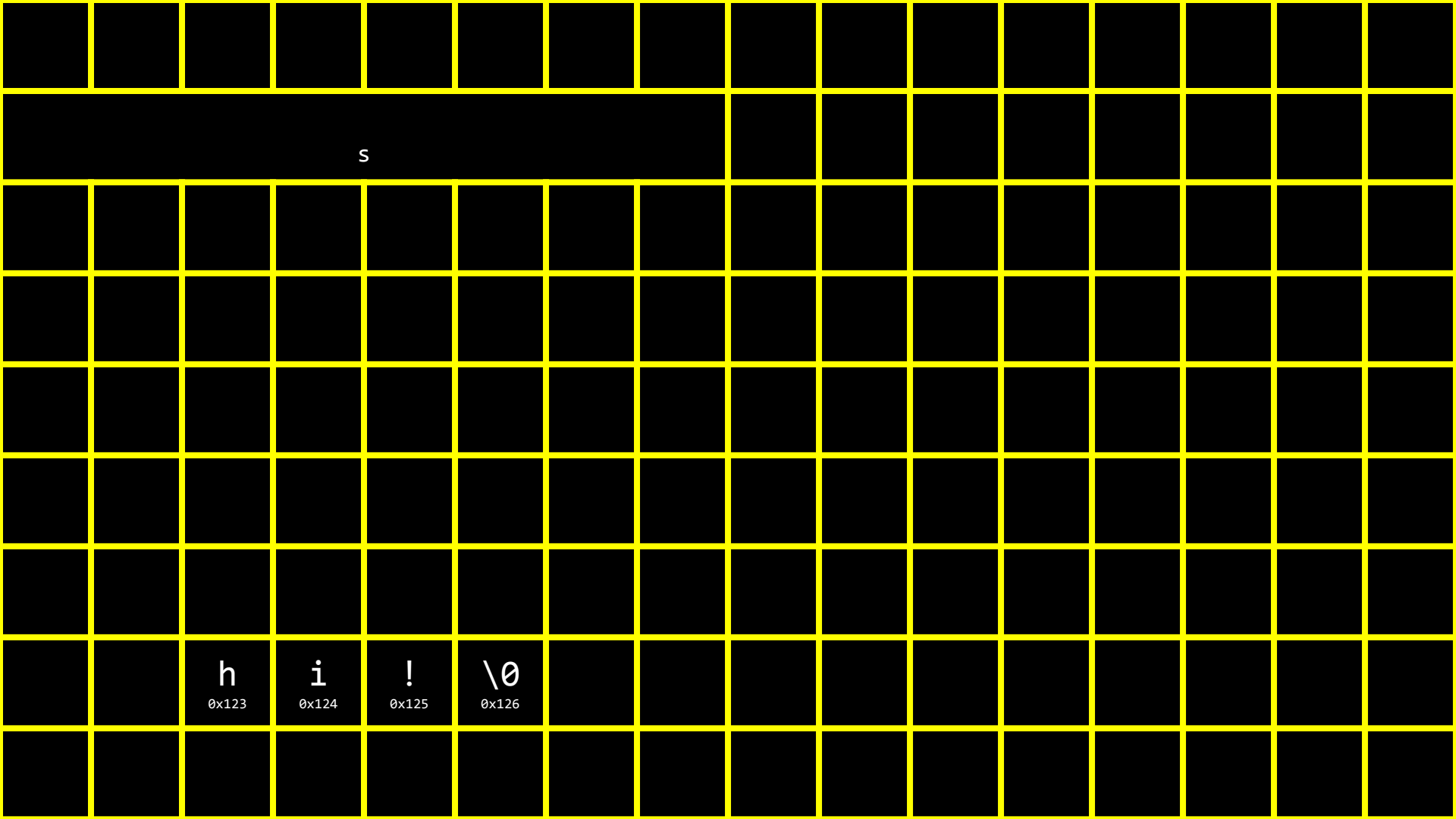
s

h

i

!

\0



0x123
s

0x123
s

 0x124	 0x125
--	--

 0x124	 0x125
--	--

!	\0
0x125	0x126

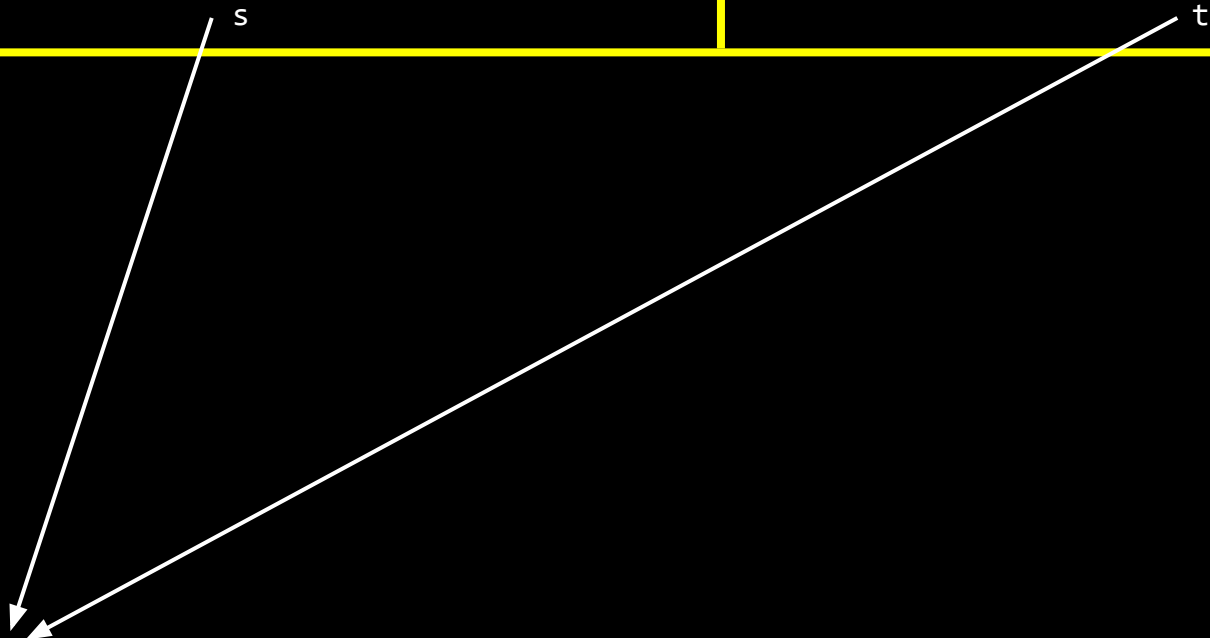
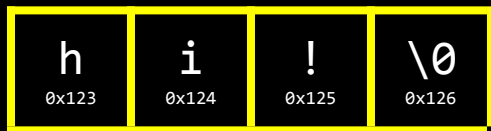
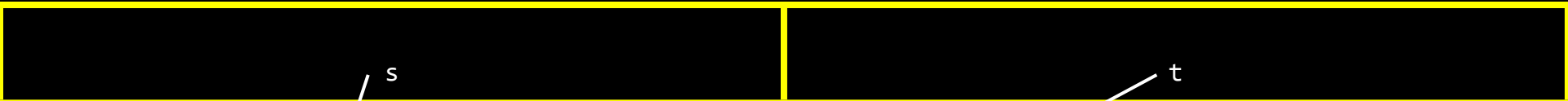
!	\0
0x125	0x126

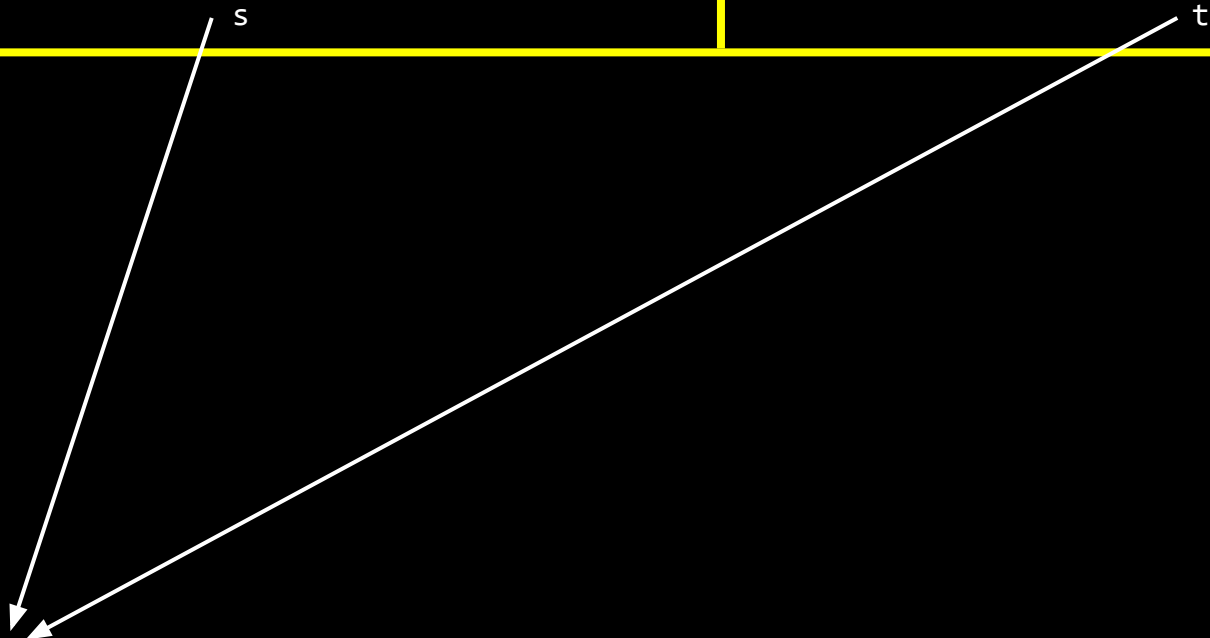
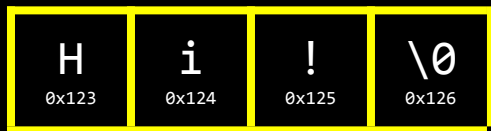
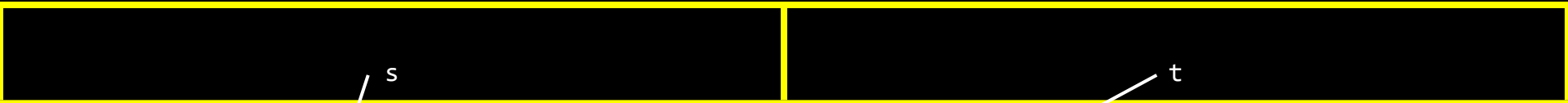
	\0	
	0x126	

	\0	
	0x126	

[illegible]

[illegible]





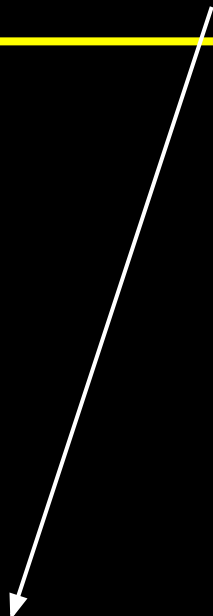
malloc

free

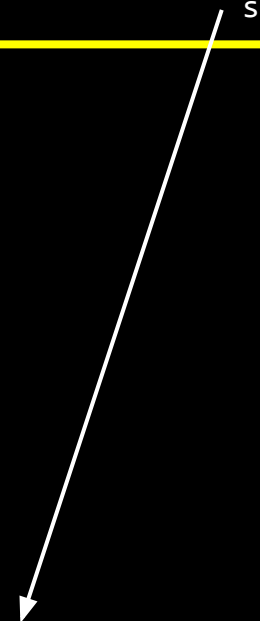
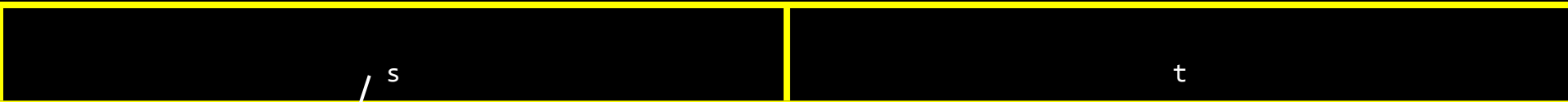
...



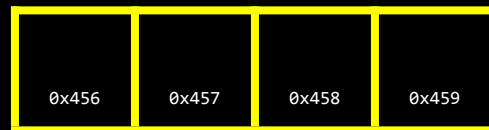
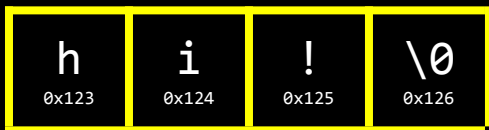
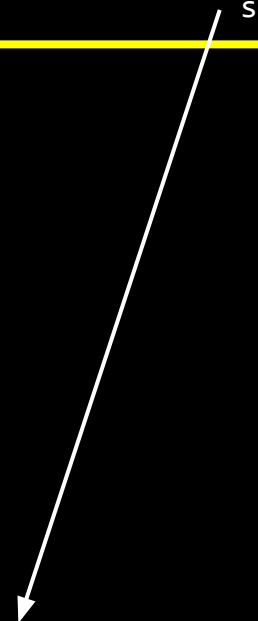
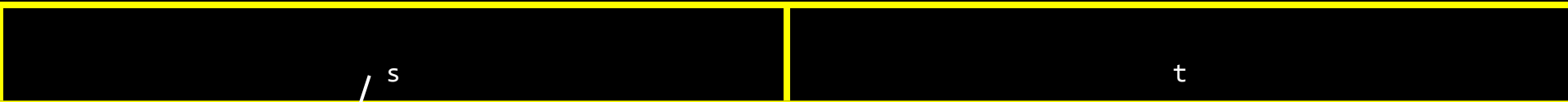
s

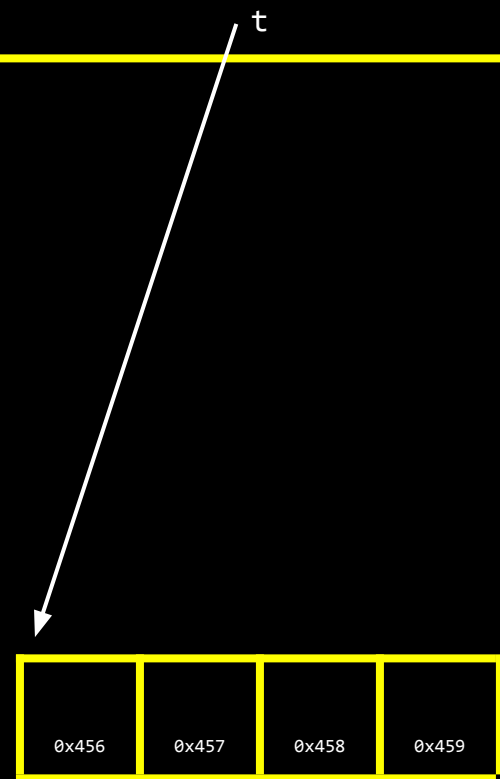
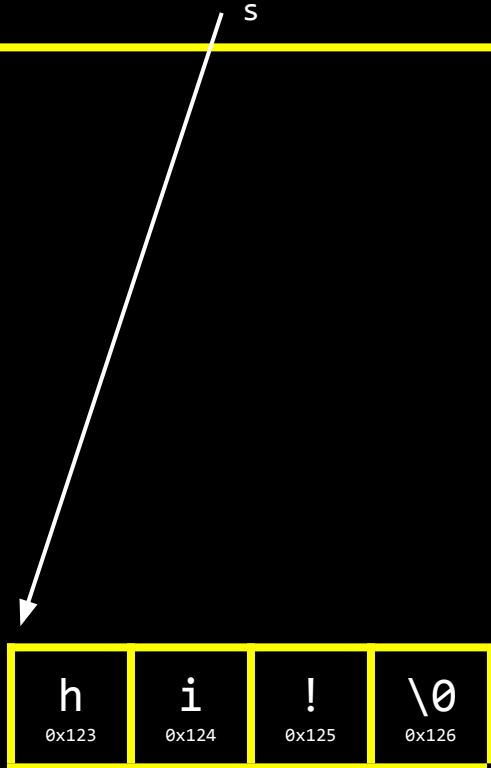
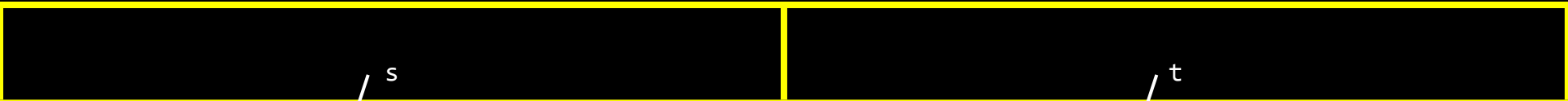


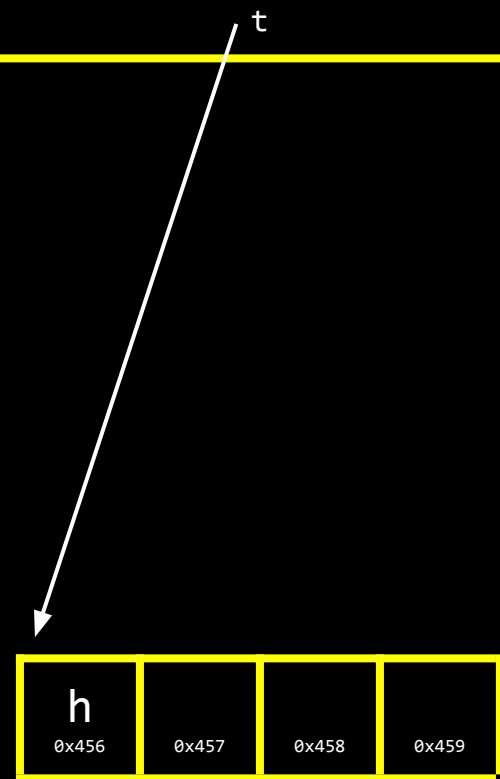
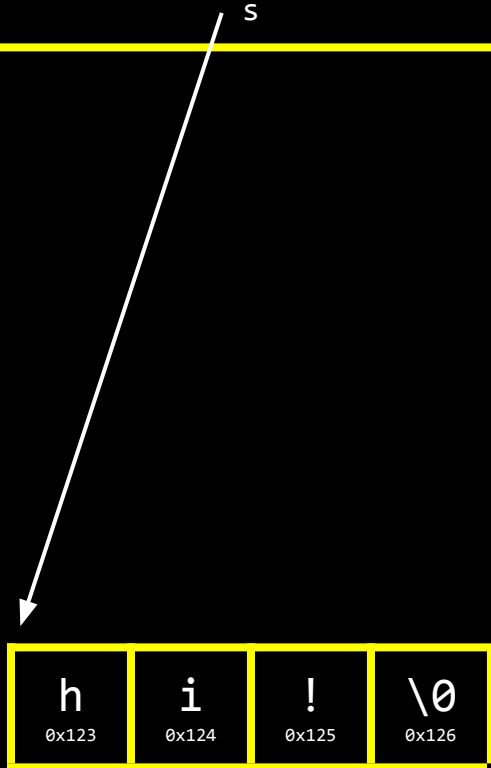
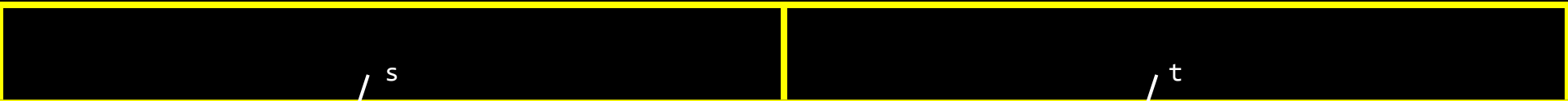
h	i	!	\0
0x123	0x124	0x125	0x126

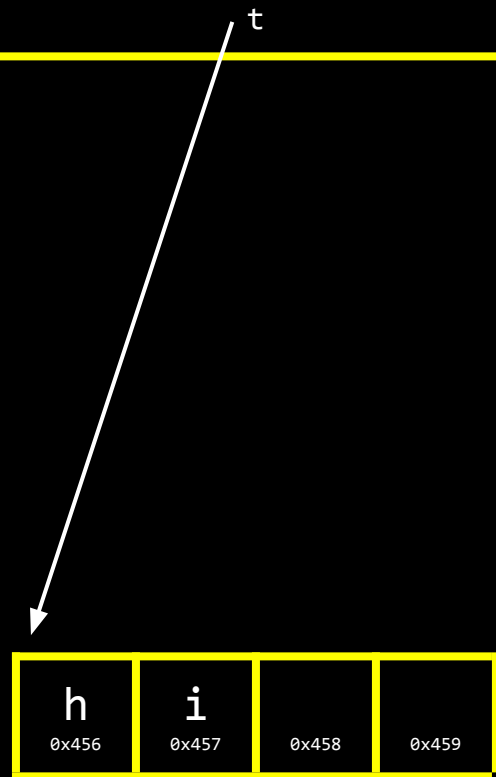
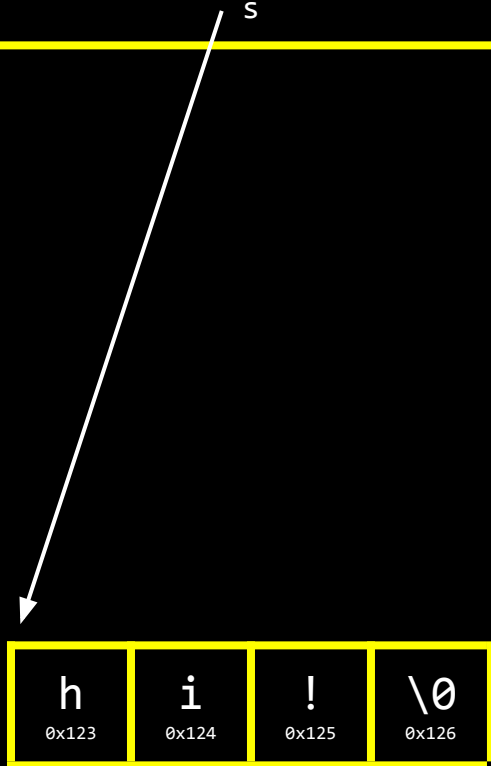
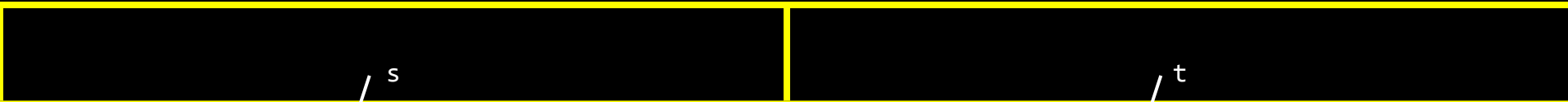


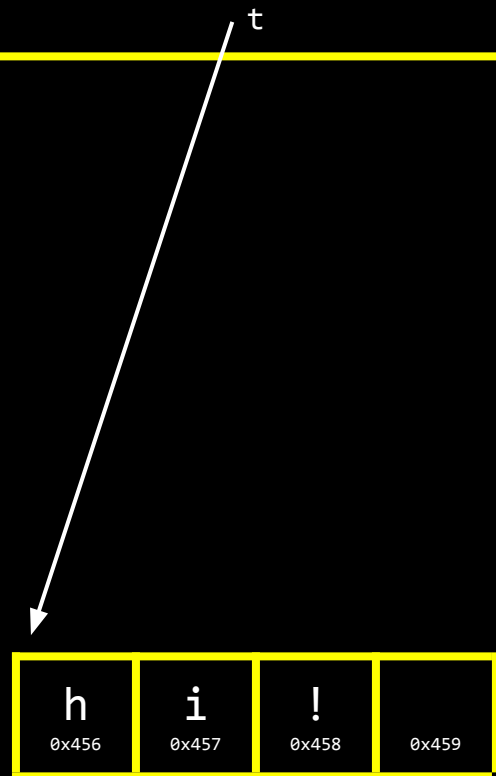
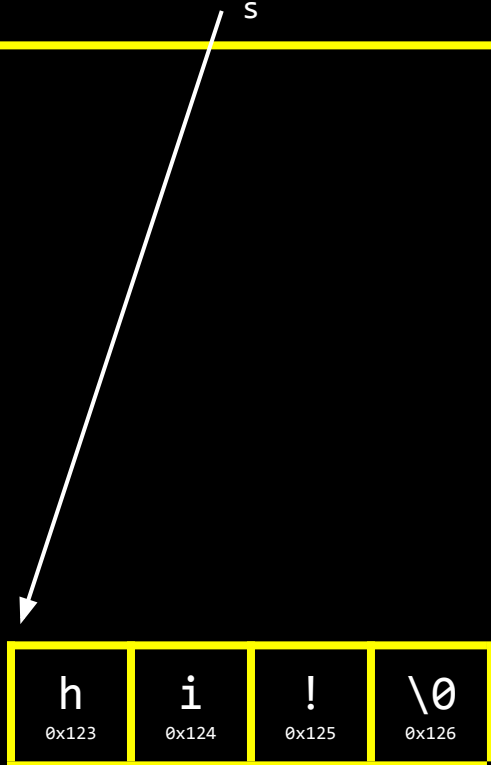
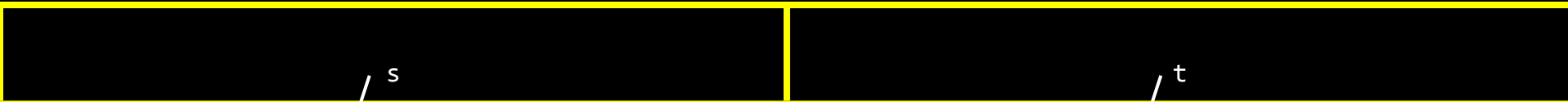
h	i	!	\0
0x123	0x124	0x125	0x126

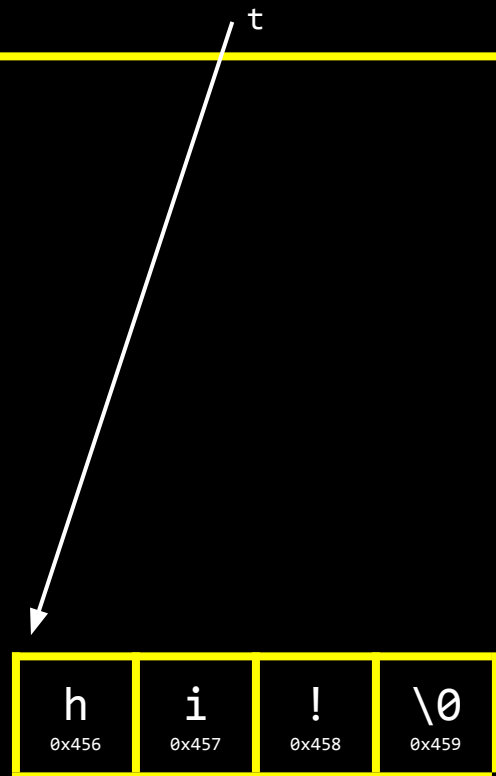
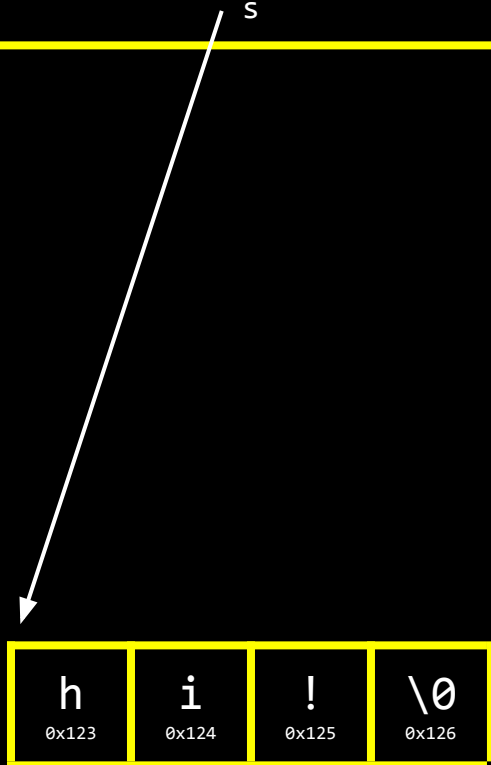
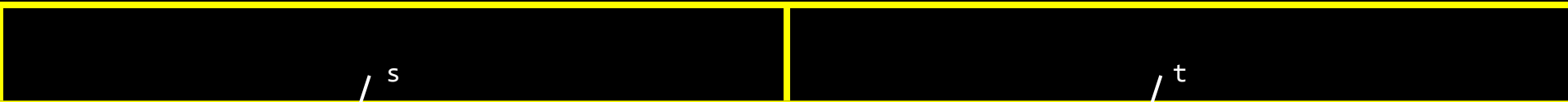


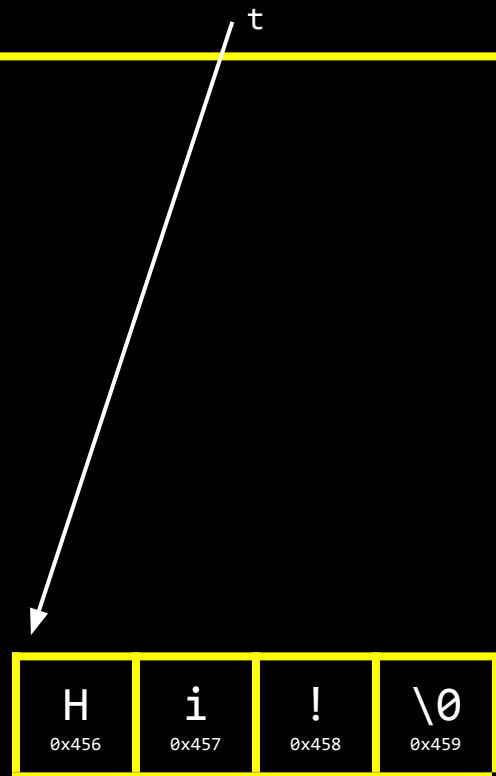
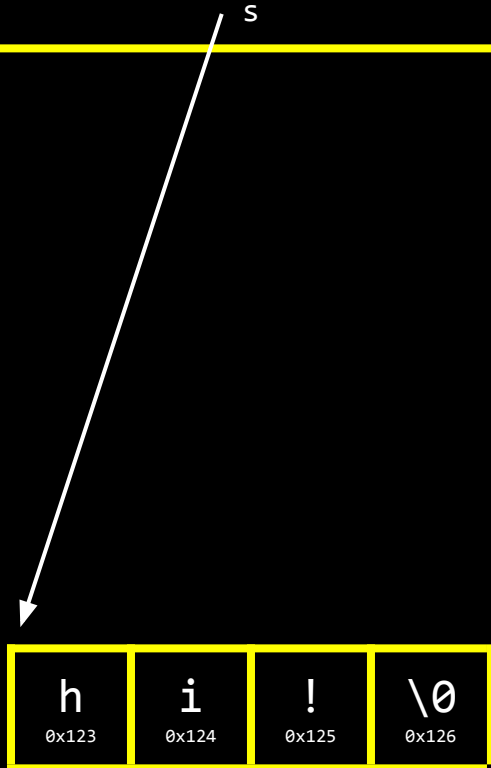
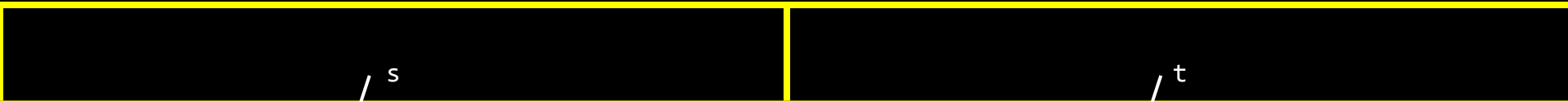












NULL

valgrind

garbage values

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;

    y = x;

    *y = 13;
}
```



```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;

    y = x;

    *y = 13;
}
```

```
int main(void)
{
    int *x;
    int *y;

    x = malloc(sizeof(int));

    *x = 42;

    y = x;

    *y = 13;
}
```



```
*y = 13;
```

```
void swap(int a, int b)
{

}
}
```

```
void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```


scope





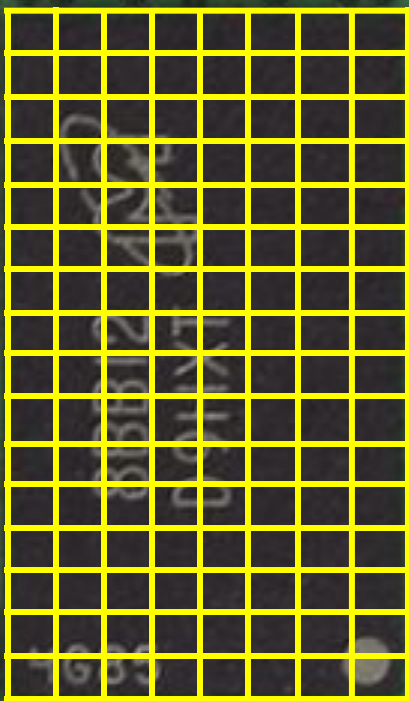
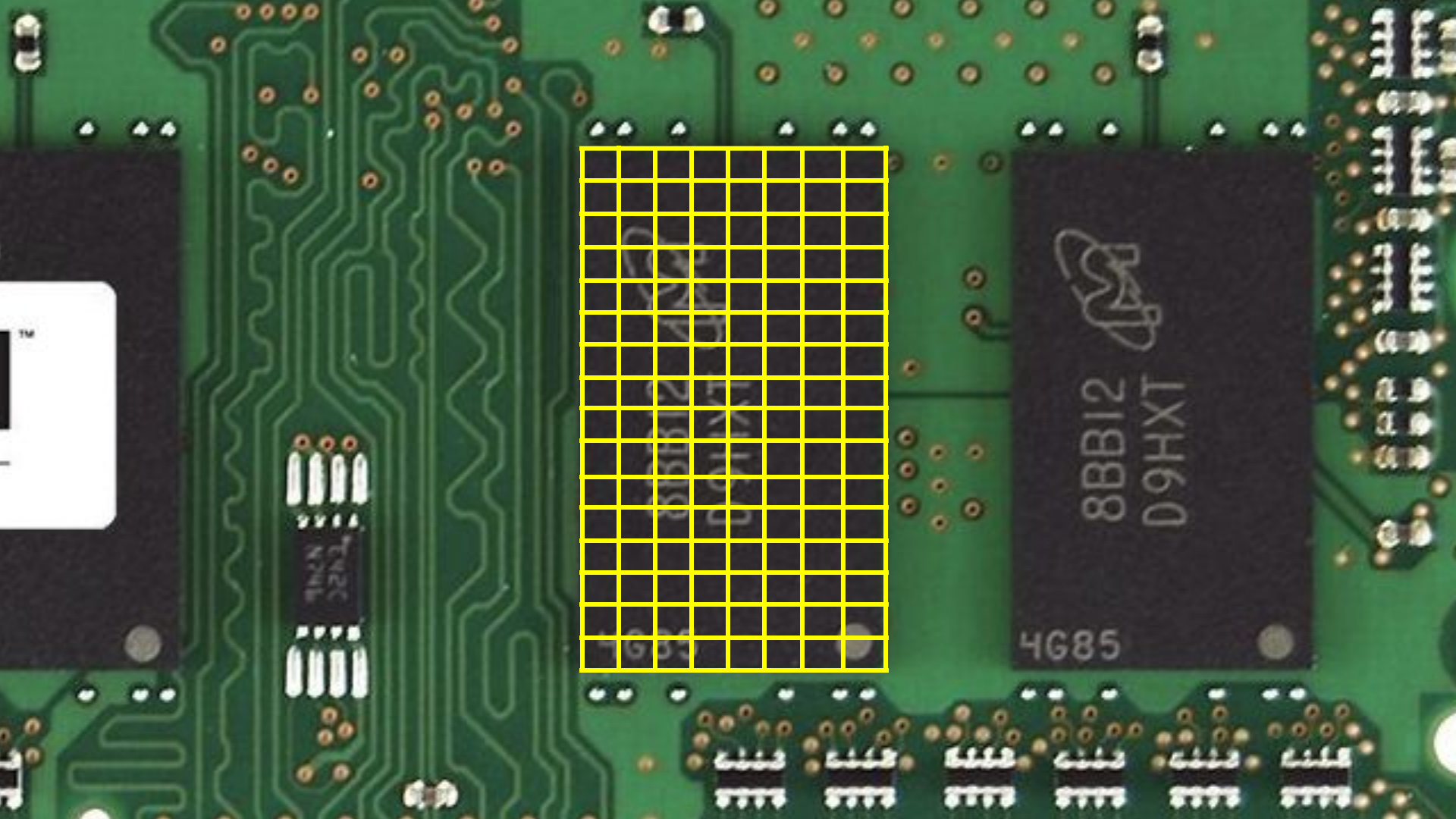
8BB12
D9HXT

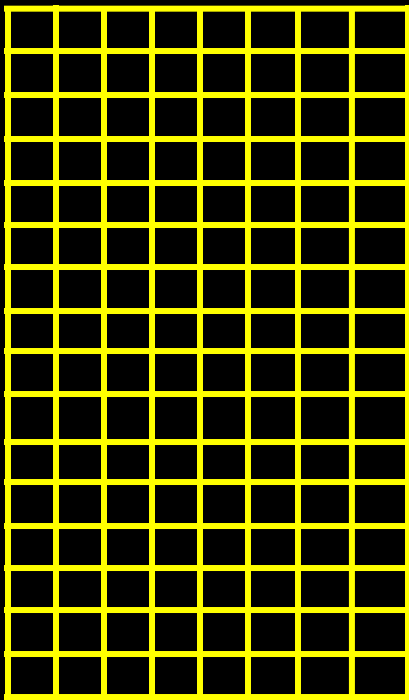
4G85

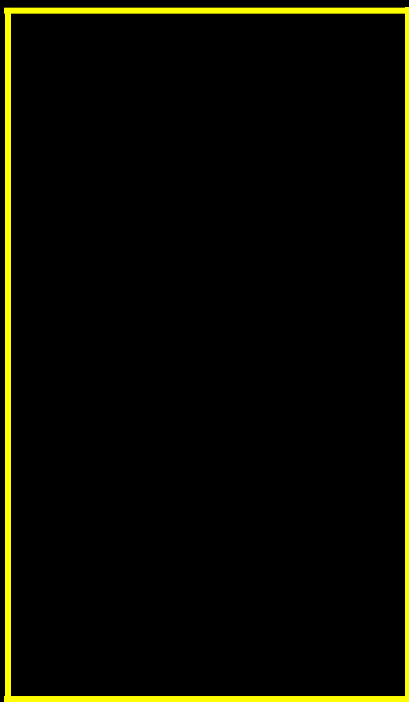


8BB12
D9HXT

4G85







machine code

A diagram of a memory block, represented as a vertical rectangle with a yellow border. The rectangle is divided into two horizontal sections. The top section is a smaller rectangle containing the text 'machine code' in white. The bottom section is a larger, empty rectangle, also outlined in yellow.

machine code

globals



A diagram showing a vertical stack of three memory regions. The top region is labeled 'machine code', the middle region is labeled 'globals', and the bottom region is labeled 'heap'. The regions are separated by horizontal lines, and the entire stack is enclosed in a yellow border.

machine code
globals
heap

machine code

globals

heap

machine code

globals

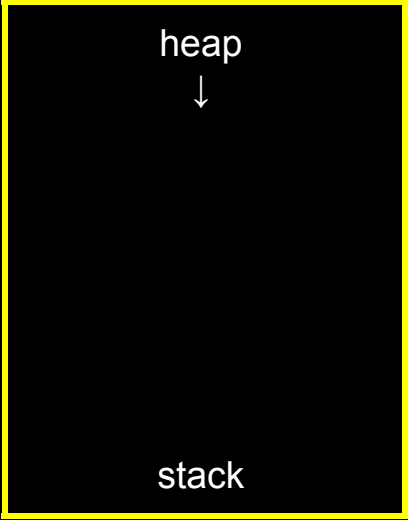
heap



machine code

globals

A diagram illustrating the memory layout. It consists of a large black rectangle with a yellow border. At the top center, the word "heap" is written in white. Below it, a white arrow points downwards. At the bottom center, the word "stack" is written in white.



A diagram illustrating the memory layout. It consists of a large black rectangle with a yellow border. At the top center, the word "heap" is written in white. Below it, a white arrow points downwards. At the bottom center, the word "stack" is written in white.

machine code

globals

A diagram illustrating memory layout. It features a large black rectangle representing memory, bounded by a yellow border. At the top center, the word "heap" is written in white. A white arrow points downwards from "heap". At the bottom center, the word "stack" is written in white. A white arrow points upwards from "stack". The arrows indicate that the heap and stack are growing towards each other.

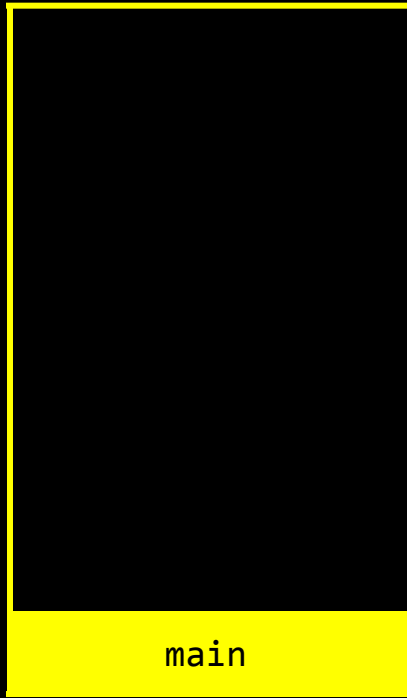


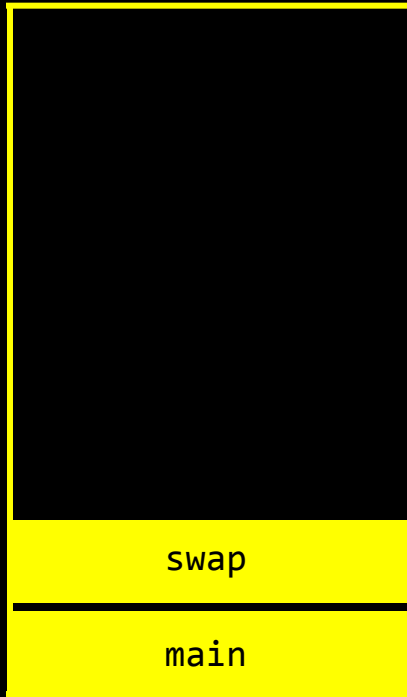
↑
stack

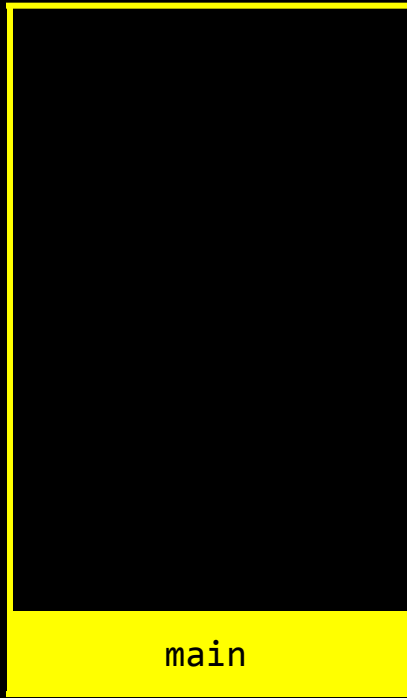


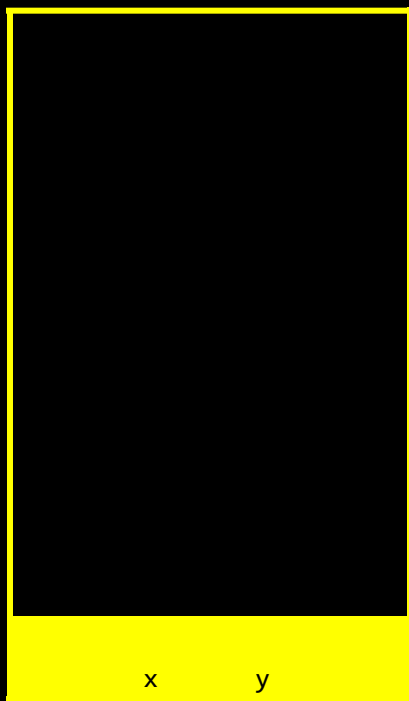


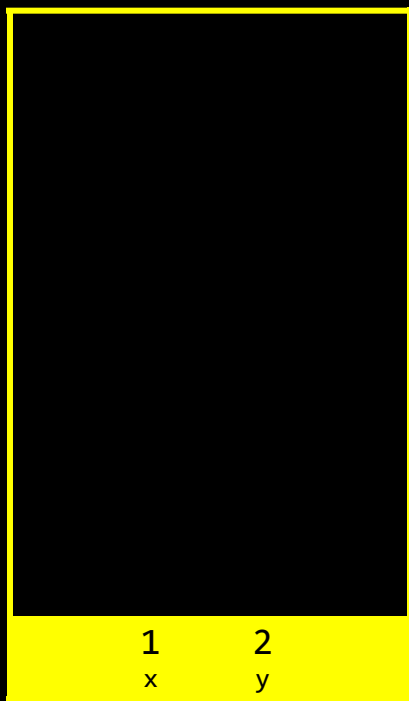
↑
stack

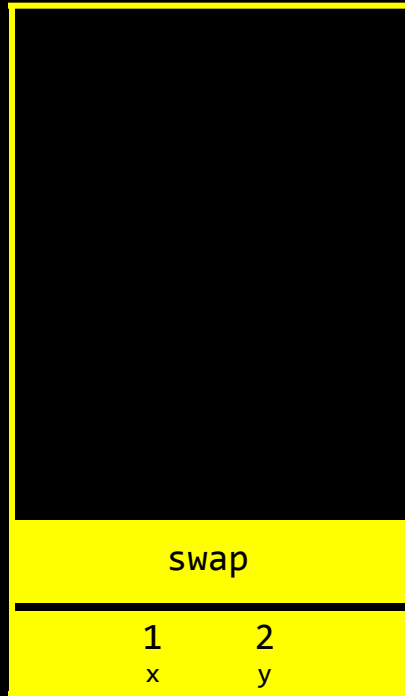




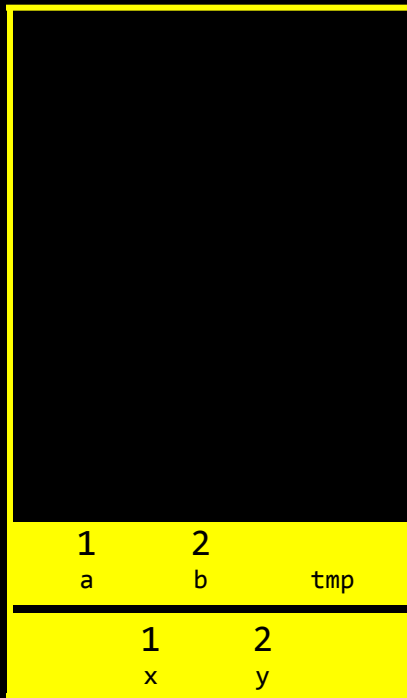




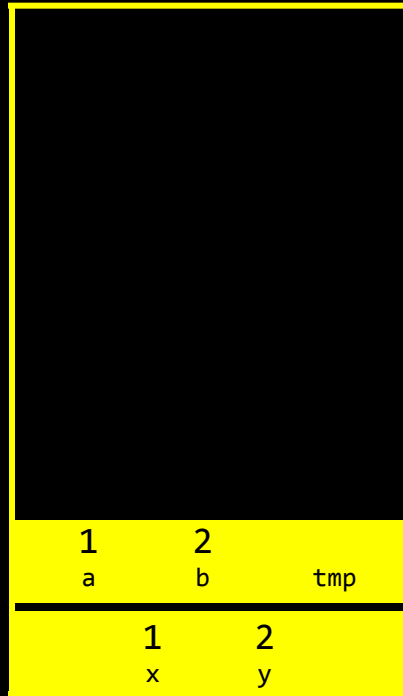




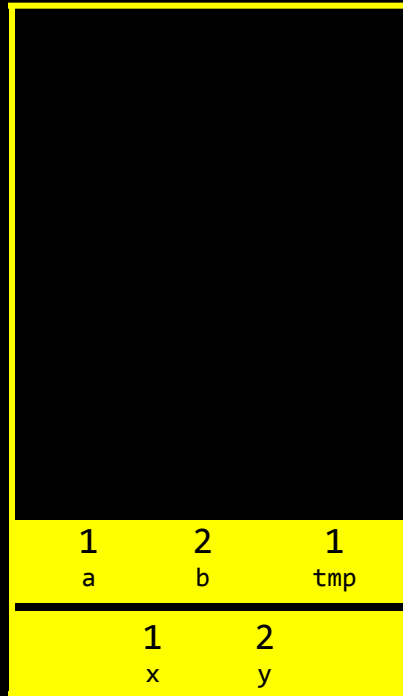
a	b	tmp
1	2	
x	y	



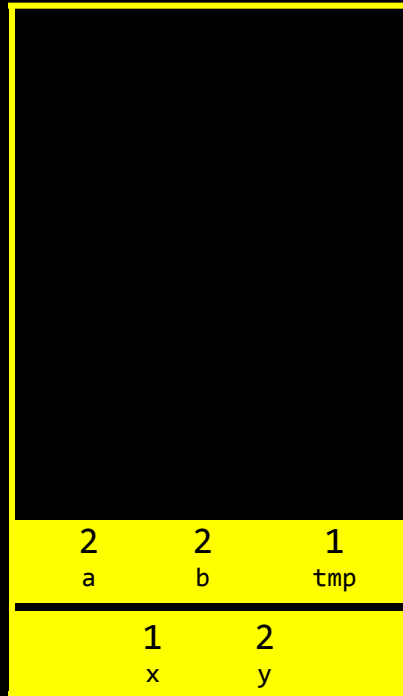
```
int tmp = a;  
a = b;  
b = tmp;
```



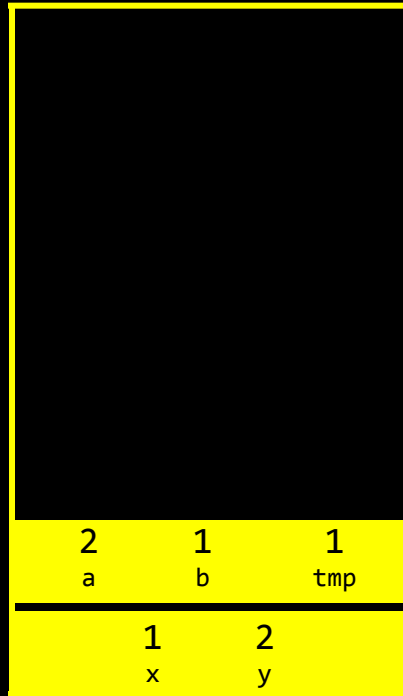
```
int tmp = a;  
a = b;  
b = tmp;
```

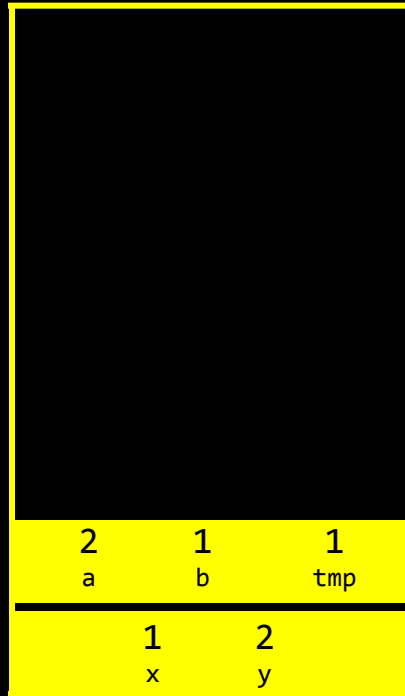


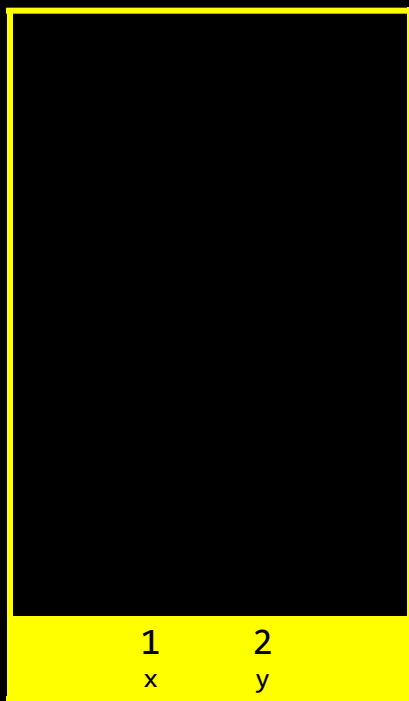
```
int tmp = a;  
a = b;  
b = tmp;
```



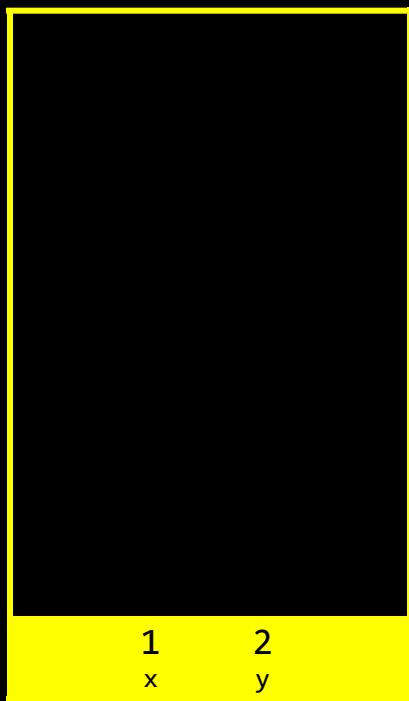
```
int tmp = a;  
a = b;  
b = tmp;
```

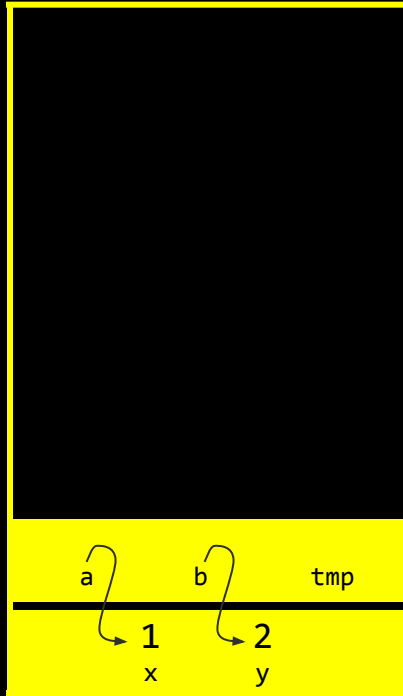




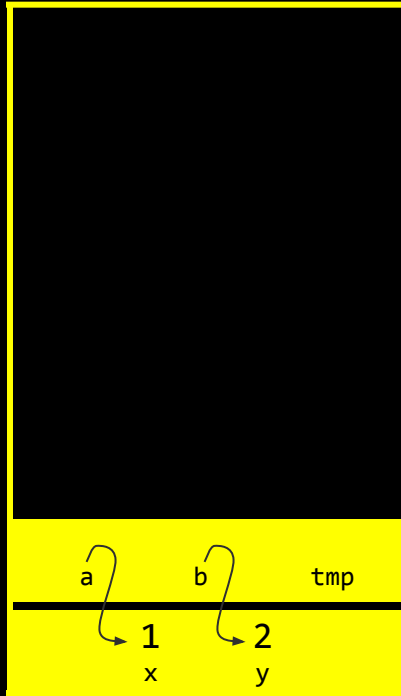



```
void swap(int *a, int *b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
```

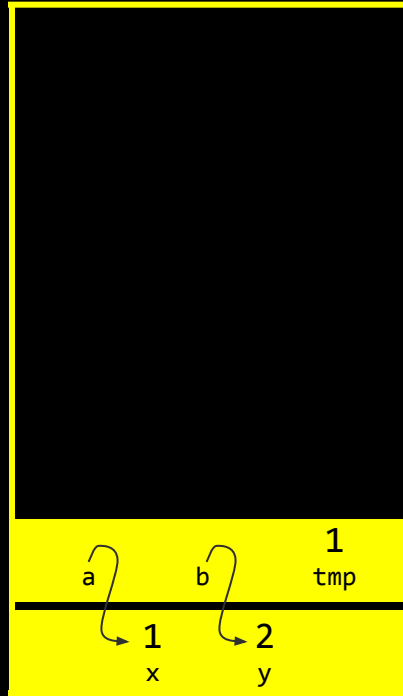




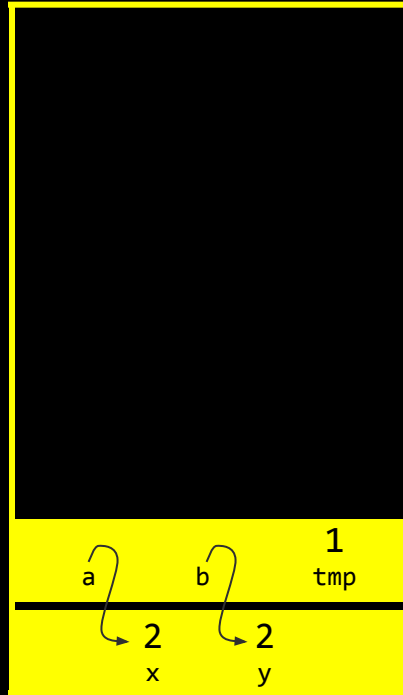
```
int tmp = *a;  
*a = *b;  
*b = tmp;
```



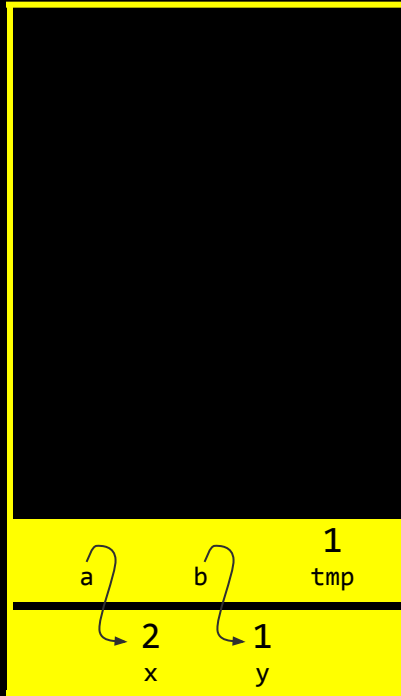
```
int tmp = *a;  
*a = *b;  
*b = tmp;
```

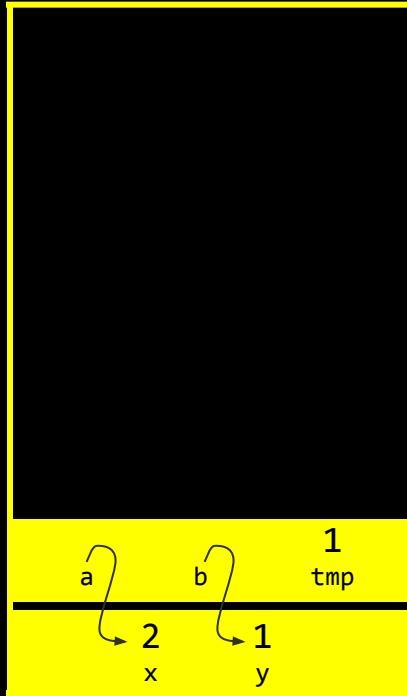


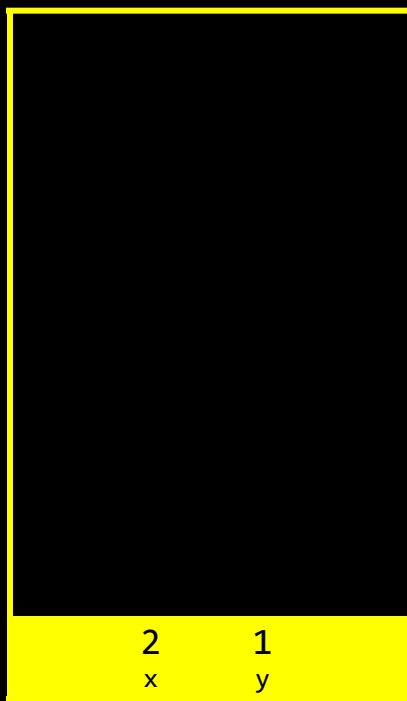

```
int tmp = *a;  
*a = *b;  
*b = tmp;
```



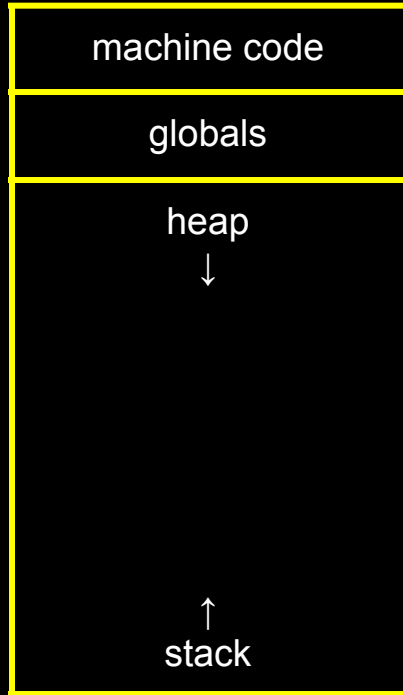
```
int tmp = *a;  
*a = *b;  
*b = tmp;
```








```
void swap(int *a, int *b)
{
    int tmp = *a;
    *a = *b;
    *b = tmp;
}
```



heap



↑
stack

heap overflow

stack overflow

buffer overflow

get_char

get_double

get_float

get_int

get_long

get_string

...

scanf

...

file I/O

BMP













This is CS50