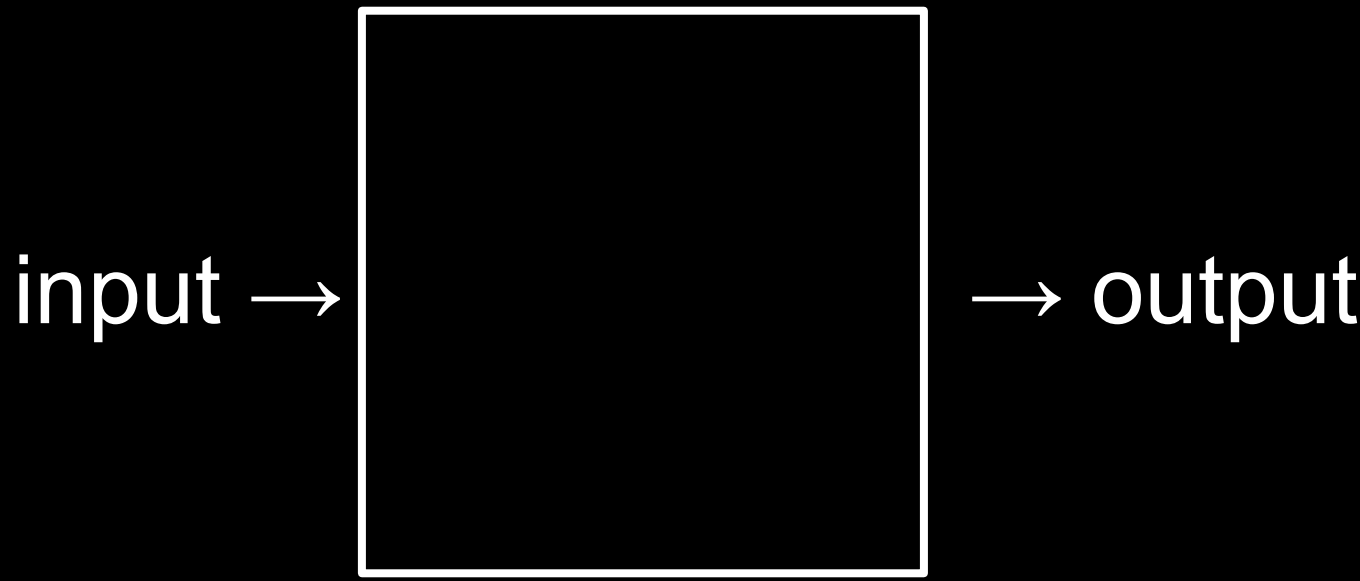
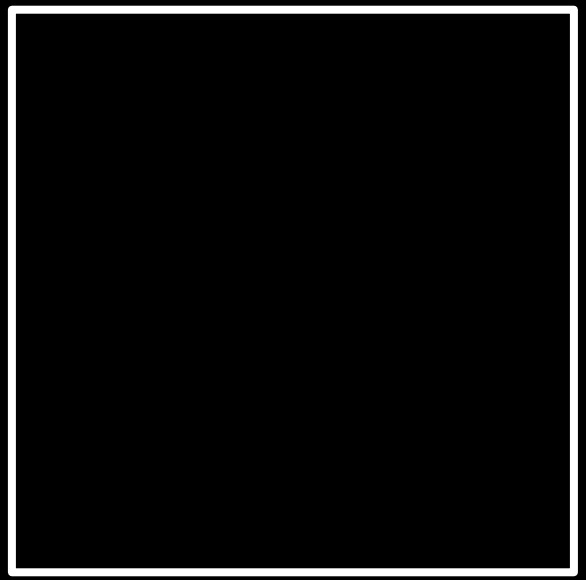
```c
#include <stdio.h>

int main(void)
{
    printf("hello, world");
}
```

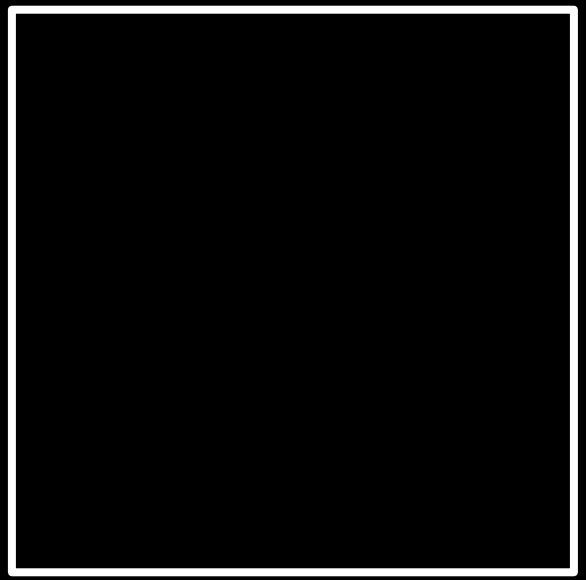
- functions
- conditions
- Boolean expressions
- loops
- ...

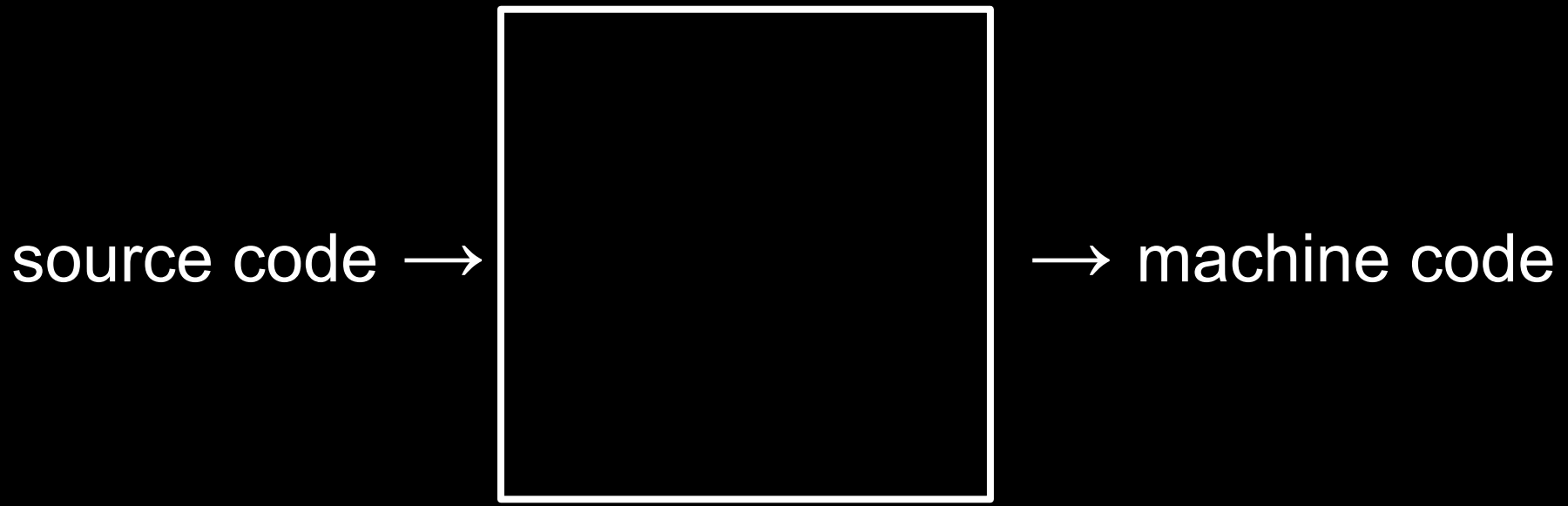
```
01111111 01000101 01001100 01000110 00000010 00000001 00000001 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000010 00000000 00111110 00000000 00000001 00000000 00000000 00000000
10110000 00000101 01000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
11010000 00010011 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 01000000 00000000 00111000 00000000
00001001 00000000 01000000 00000000 00100100 00000000 00100001 00000000
00000110 00000000 00000000 00000000 00000101 00000000 00000000 00000000
01000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 01000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 01000000 00000000 00000000 00000000 00000000 00000000
11111000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
11111000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
00001000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000011 00000000 00000000 00000000 00000100 00000000 00000000 00000000
00111000 00000010 00000000 00000000 00000000 00000000 00000000 00000000
...
```

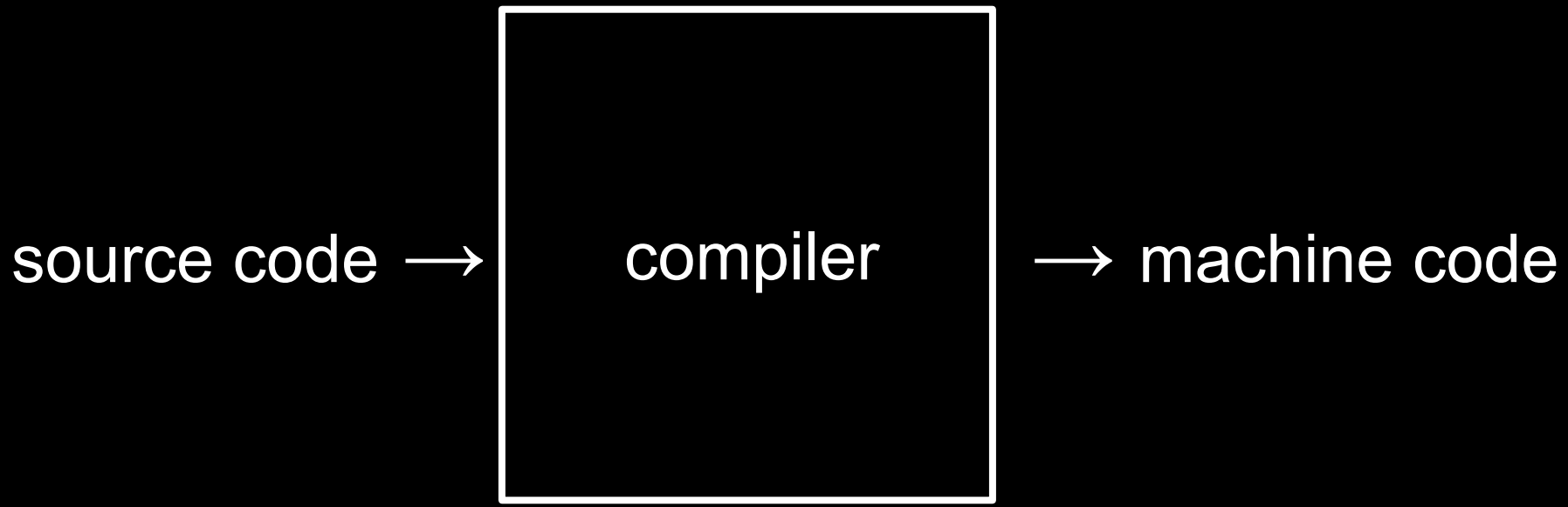

input →
→ output



source code ⟶

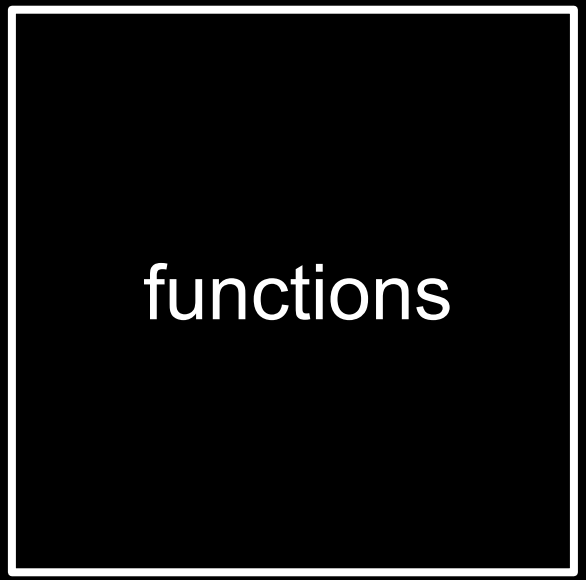


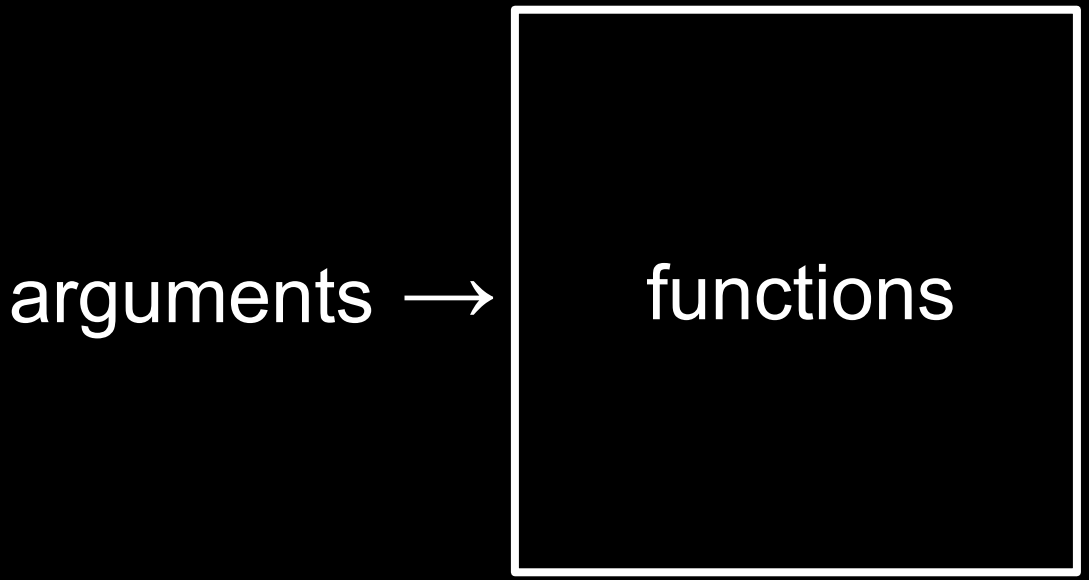

source code ⟶
⟶ machine code


source code
compiler
machine code

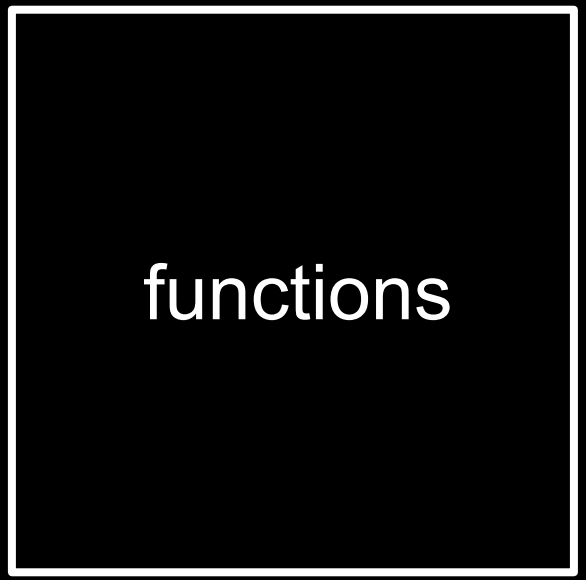
```
make hello

./hello
```

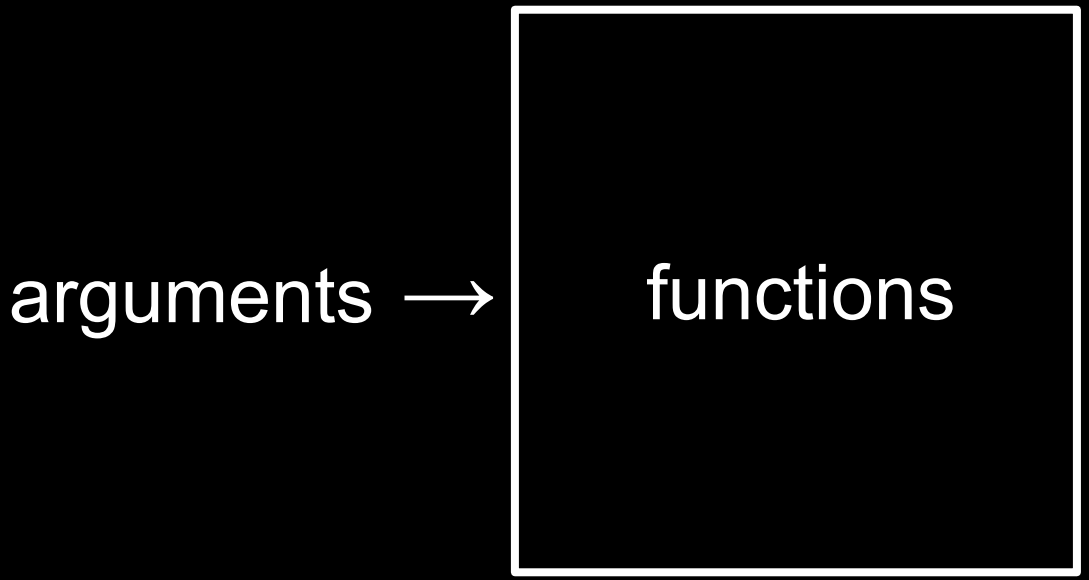
functions, arguments

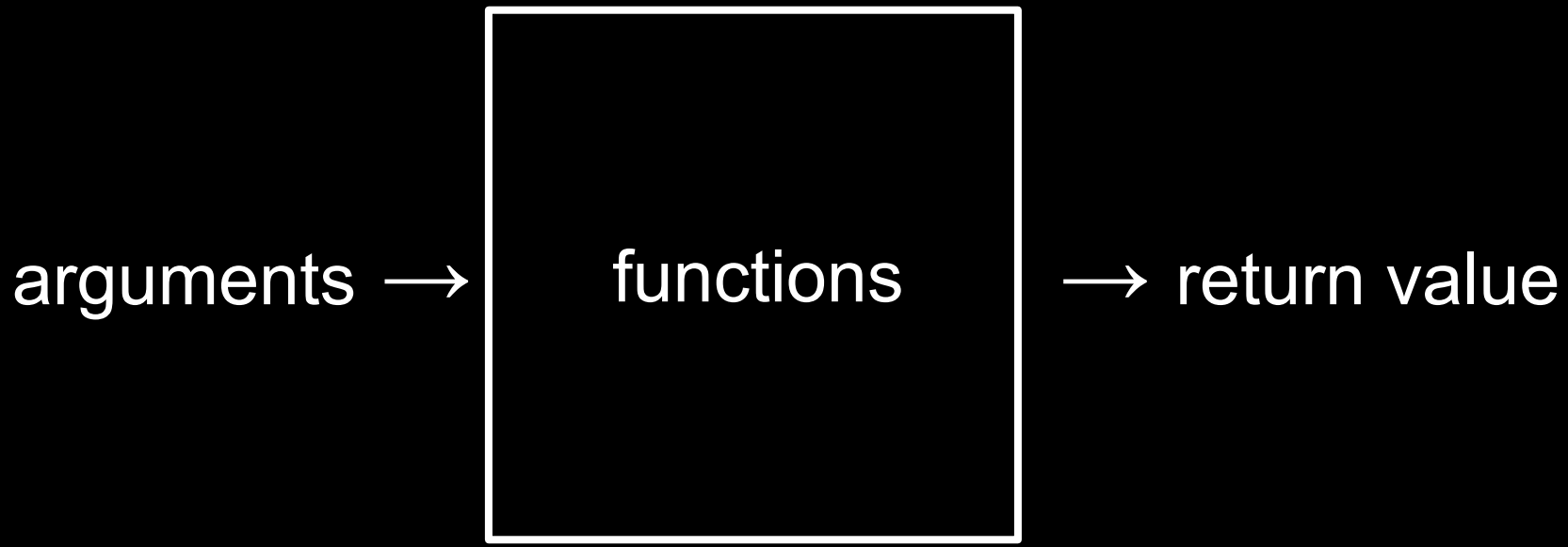

functions


arguments ⟶
functions

return values, variables


functions


arguments ⟶
functions


arguments
functions
return value

main

header files

types

```
bool

char

double

float

int

long

string

...
```

format codes

%c

%f

%i

%li

%s

```
%c  char

%f  float, double

%i  int

%li long

%s  string
```

operators

+

-

*

/

%

```
+  addition
-  subtraction
*  multiplication
/  division
%  remainder
```

variables, syntactic sugar

conditions

loops

abstraction


crucial
by Micron
8BBI2
D9HXT
8BBI2
D9HXT
4G85
4G85
4G85
47J6

floating-point imprecision

integer overflow

000

001

010

011

100

101

110

111

1000

000

1 January 2000

95

96

97

98

99

100

00

19 January 2038

```c
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

```
01111111 01000101 01001100 01000110 00000010 00000001 00000001 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000010 00000000 00111110 00000000 00000001 00000000 00000000 00000000
10110000 00000101 01000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
11010000 00010011 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 01000000 00000000 00111000 00000000
00001001 00000000 01000000 00000000 00100100 00000000 00100001 00000000
00000110 00000000 00000000 00000000 00000101 00000000 00000000 00000000
01000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 01000000 00000000 00000000 00000000 00000000 00000000
01000000 00000000 01000000 00000000 00000000 00000000 00000000 00000000
11111000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
11111000 00000001 00000000 00000000 00000000 00000000 00000000 00000000
00001000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000011 00000000 00000000 00000000 00000100 00000000 00000000 00000000
00111000 00000010 00000000 00000000 00000000 00000000 00000000 00000000
...
```

```
make hello

./hello
```

```
clang hello.c

./a.out
```

```
clang -o hello hello.c

./hello
```

```
clang -o hello hello.c -lcs50

./hello
```

```
make hello

./hello
```

compiling

preprocessing

compiling

assembling

linking

preprocessing

compiling

assembling

linking

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name? ");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name? ");
    printf("hello, %s\n", name);
}
```

```
string get_string(string prompt);
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name? ");
    printf("hello, %s\n", name);
}
```

```
string get_string(string prompt);
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name? ");
    printf("hello, %s\n", name);
}
```

```
string get_string(string prompt);
int printf(string format, ...);


int main(void)
{
    string name = get_string("What's your name? ");
    printf("hello, %s\n", name);
}
```

```
...
string get_string(string prompt);
int printf(string format, ...);
...

int main(void)
{
    string name = get_string("What's your name? ");
    printf("hello, %s\n", name);
}
```

preprocessing

compiling

assembling

linking

```
...
string get_string(string prompt);
int printf(string format, ...);
...

int main(void)
{
    string name = get_string("What's your name? ");
    printf("hello, %s\n", name);
}
```

```
...
main:                                   # @main
    .cfi_startproc
# BB#0:
    pushq    %rbp
.Ltmp0:
    .cfi_def_cfa_offset 16
.Ltmp1:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp2:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    xorl    %eax, %eax
    movl    %eax, %edi
    movabsq    $.L.str, %rsi
    movb    $0, %al
    callq    get_string
    movabsq    $.L.str.1, %rdi
    movq    %rax, -8(%rbp)
    movq    -8(%rbp), %rsi
    movb    $0, %al
    callq    printf
    ...
```

```
...
main:                                   # @main
    .cfi_startproc
# BB#0:
    pushq    %rbp
.Ltmp0:
    .cfi_def_cfa_offset 16
.Ltmp1:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp2:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    xorl    %eax, %eax
    movl    %eax, %edi
    movabsq    $.L.str, %rsi
    movb    $0, %al
    callq    get_string
    movabsq    $.L.str.1, %rdi
    movq    %rax, -8(%rbp)
    movq    -8(%rbp), %rsi
    movb    $0, %al
    callq    printf
    ...
```

```
...
main:                                   # @main
    .cfi_startproc
# BB#0:
    pushq    %rbp
.Ltmp0:
    .cfi_def_cfa_offset 16
.Ltmp1:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp2:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    xorl    %eax, %eax
    movl    %eax, %edi
    movabsq    $.L.str, %rsi
    movb    $0, %al
    callq    get_string
    movabsq    $.L.str.1, %rdi
    movq    %rax, -8(%rbp)
    movq    -8(%rbp), %rsi
    movb    $0, %al
    callq    printf
    ...
```

preprocessing

compiling

assembling

linking

```
...
main:                                   # @main
    .cfi_startproc
# BB#0:
    pushq    %rbp
.Ltmp0:
    .cfi_def_cfa_offset 16
.Ltmp1:
    .cfi_offset %rbp, -16
    movq    %rsp, %rbp
.Ltmp2:
    .cfi_def_cfa_register %rbp
    subq    $16, %rsp
    xorl    %eax, %eax
    movl    %eax, %edi
    movabsq    $.L.str, %rsi
    movb    $0, %al
    callq    get_string
    movabsq    $.L.str.1, %rdi
    movq    %rax, -8(%rbp)
    movq    -8(%rbp), %rsi
    movb    $0, %al
    callq    printf
    ...
```

```
01111111010001010100110001000110
00000010000000010000000100000000
00000000000000000000000000000000
00000000000000000000000000000000
00000001000000000011111000000000
00000001000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
10100000000000100000000000000000
00000000000000000000000000000000
00000000000000000000000000000000
01000000000000000000000000000000
00000000000000000100000000000000
00001010000000000000000100000000
01010101010010001000100111100101
01001000100000111110110000010000
00110001110000001000100111000111
01001000101111100000000000000000
00000000000000000000000000000000
00000000000000001011000000000000
11101000000000000000000000000000
00000000010010001011111100000000
00000000000000000000000000000000
00000000000000000000000001001000
...
```

preprocessing

compiling

assembling

linking

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name? ");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name? ");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name? ");
    printf("hello, %s\n", name);
}
```

```c
#include <cs50.h>
#include <stdio.h>


int main(void)
{
    string name = get_string("What's your name? ");
    printf("hello, %s\n", name);
}
```

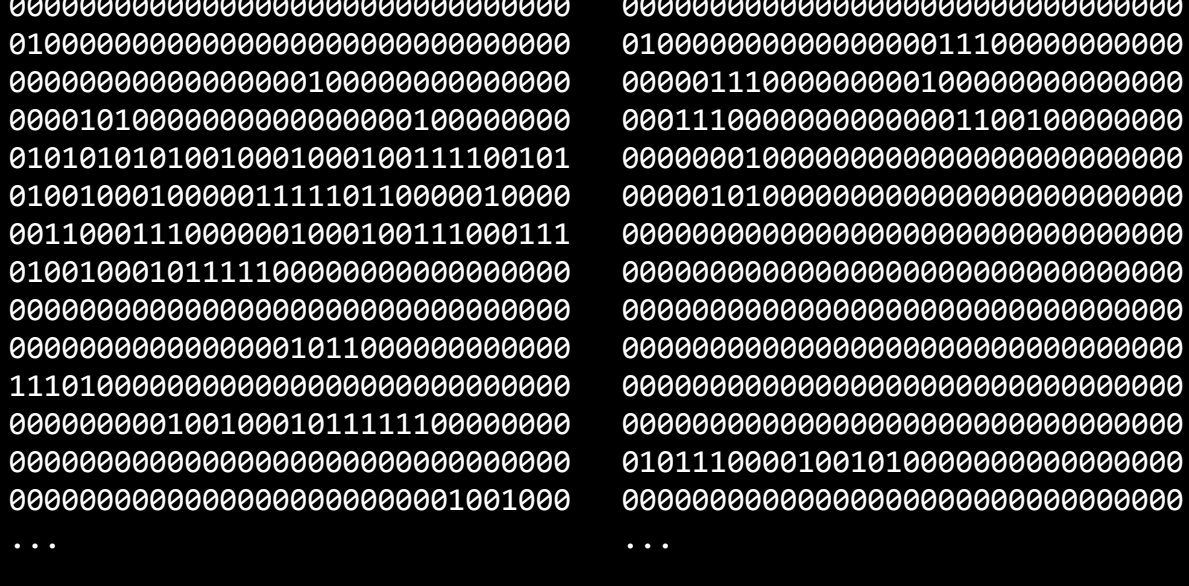
hello.c

hello.c

cs50.c

hello.c cs50.c stdio.c


cs50.c
stdio.c



stdio.c

```
011111110100010101001100010001100000001000000001000000010000000000000000000000000000000000000000000000
000000000000000000000000000000000100000000001111100000000000000001000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000001010000000000010000000000000000000000000000000000000000000000000000000000000000000000000
000000000100000000000000000000000000000000000000000000000100000000000000000010100000000000000001000000
000101010101001000100010011110010101001000100000111110110000010000001100011100000010001001110001110100
100010111110000000000000000000000000000000000000000000000000000000000000000010110000000000001110100000
000000000000000000000000000000010010001011111100000000000000000000000000000000000000000000000000000000
0000000001001000...01111111010001010100110001000110000000100000000100000001000000000000000000000000000
000000000000000000000000000000000000000000000000000110000000000111110000000000000000100000000000000000
000000011000000000011110000000000000000000000000000000000000000000000000100000000000000000000000000000
000000000000000000000000000000000001010000011001000000000000000000000000000000000000000000000000000000
000000000000000000000000000010000000000000001110000000000000000011100000000010000000000000000011100000
000000001100100000000000000010000000000000000000000000000010100000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000001011100001001010000000000000
00000000000000000000000000000000000...0010111101101100011010010110001001100011001011100111001101101111
001011100011011000100000001011110111010101110011011100100010111101101100011010010110001000101111011110
000011100000110110010111110011011000110100001011010110110001101001011011100111010101111000001011010110
011101101110011101010010111101101100011010010110001001100011010111110110111001101111011011100111001101
101000011000010111001001100101011001000010111001100001001000000010000001000001010100110101111101001110
0100010101000101010001000100010101000100001000000010100000010000000101111011011000110100101100010001011
110111110000011100000110110010111110011011000110100001011010110110001101001011011100111010101111000001 0
101101011110000011100000110110001011010011011000110100...
```

preprocessing

compiling

assembling

linking

compiling

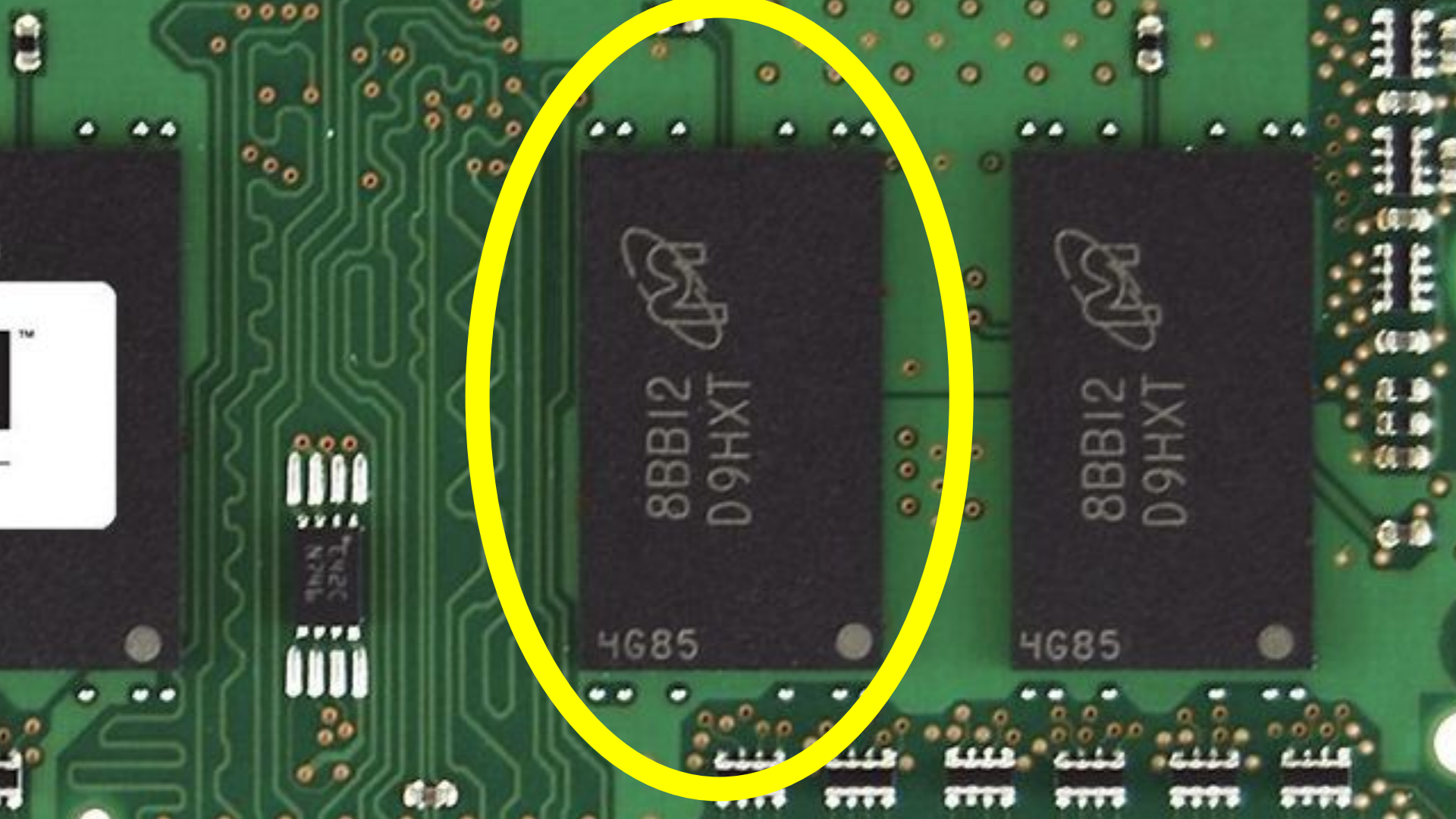
types

```
bool

char

double

float

int

long

string

...
```

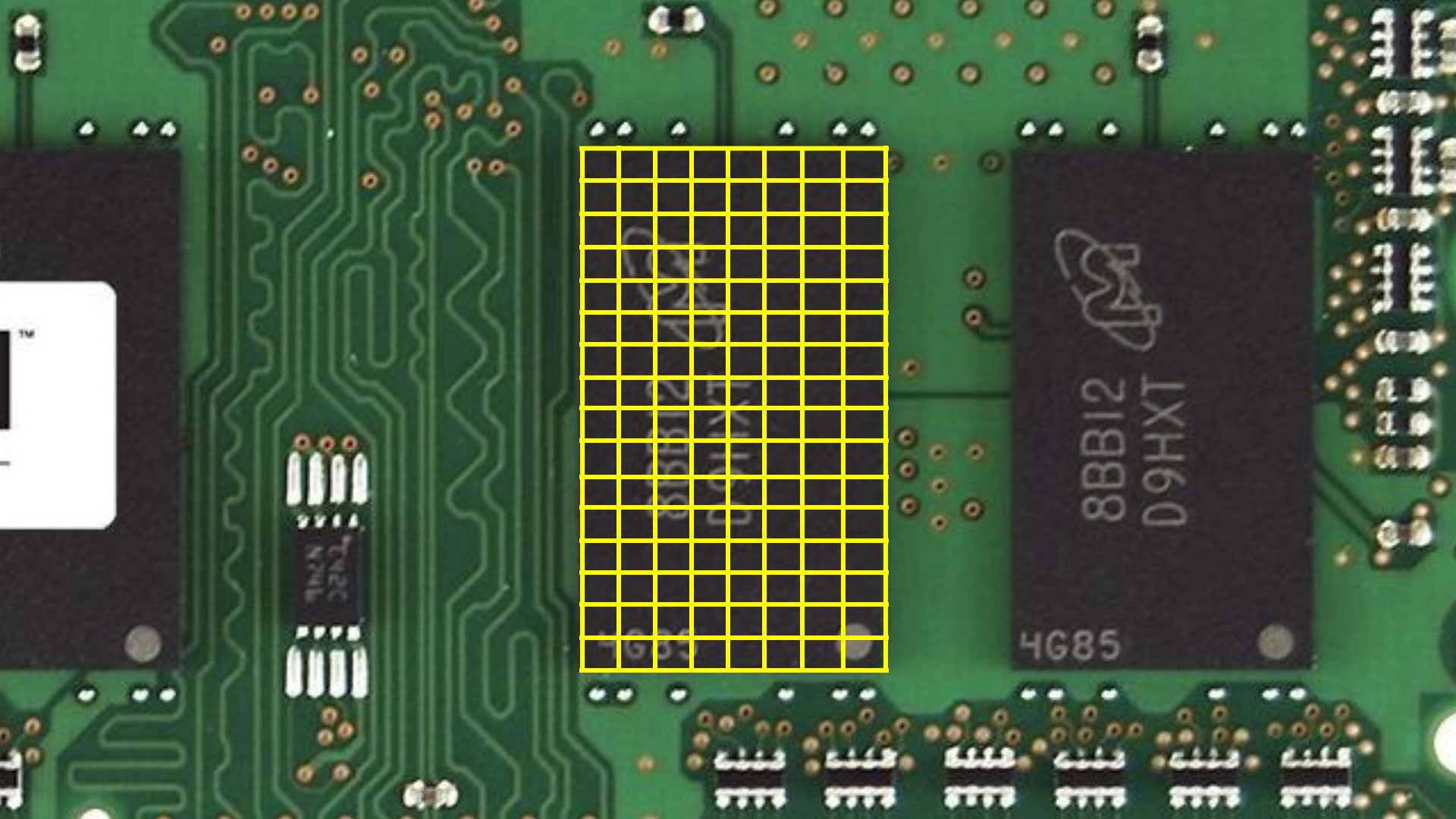
`bool`	1 byte
`char`	1 byte
`double`	8 bytes
`float`	4 bytes
`int`	4 bytes
`long`	8 bytes
`string`	? bytes
...	

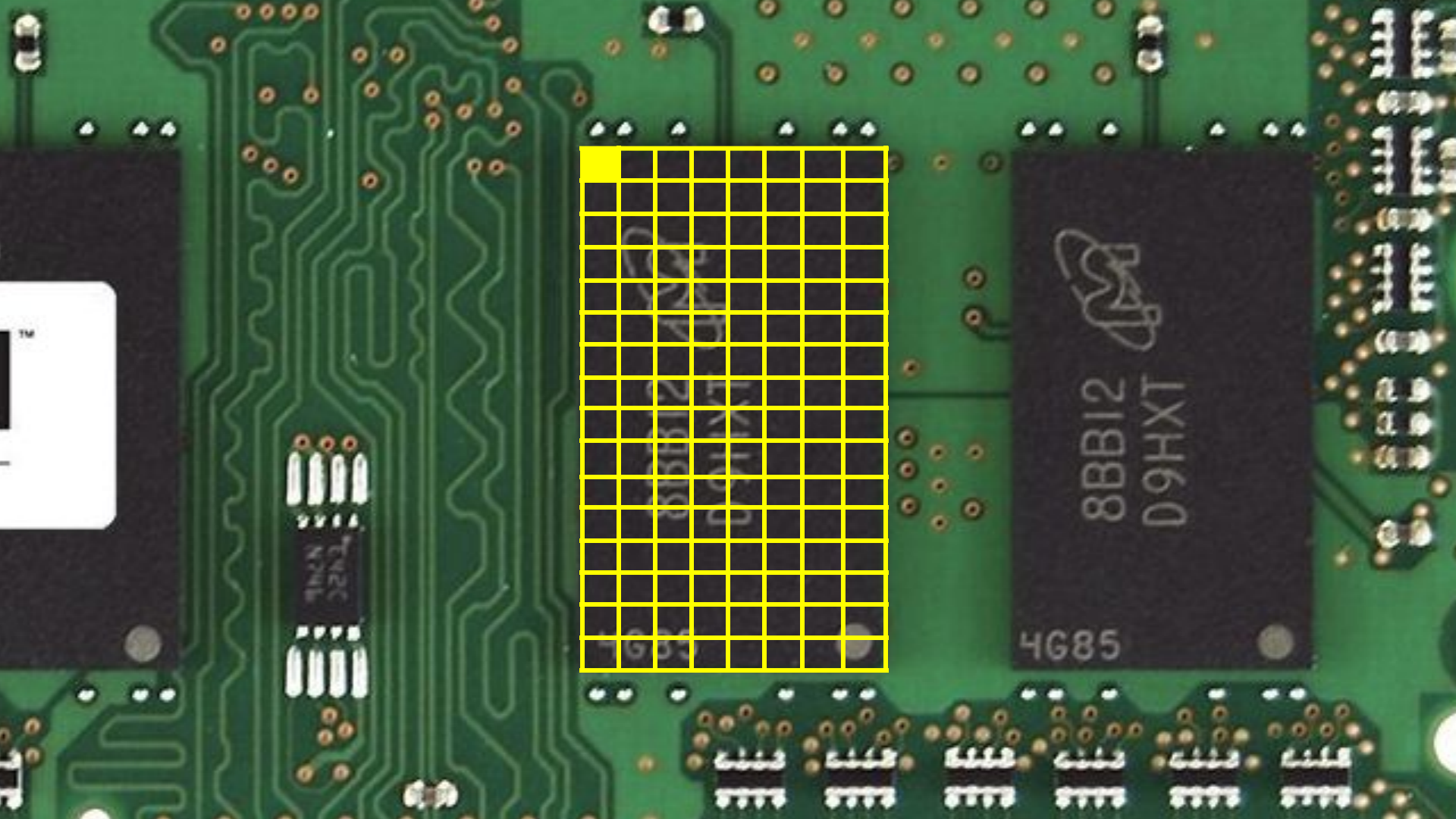

crucial
by Micron
8BBI2
D9HXT
8BBI2
D9HXT
4G85
4G85
4G85
47J6

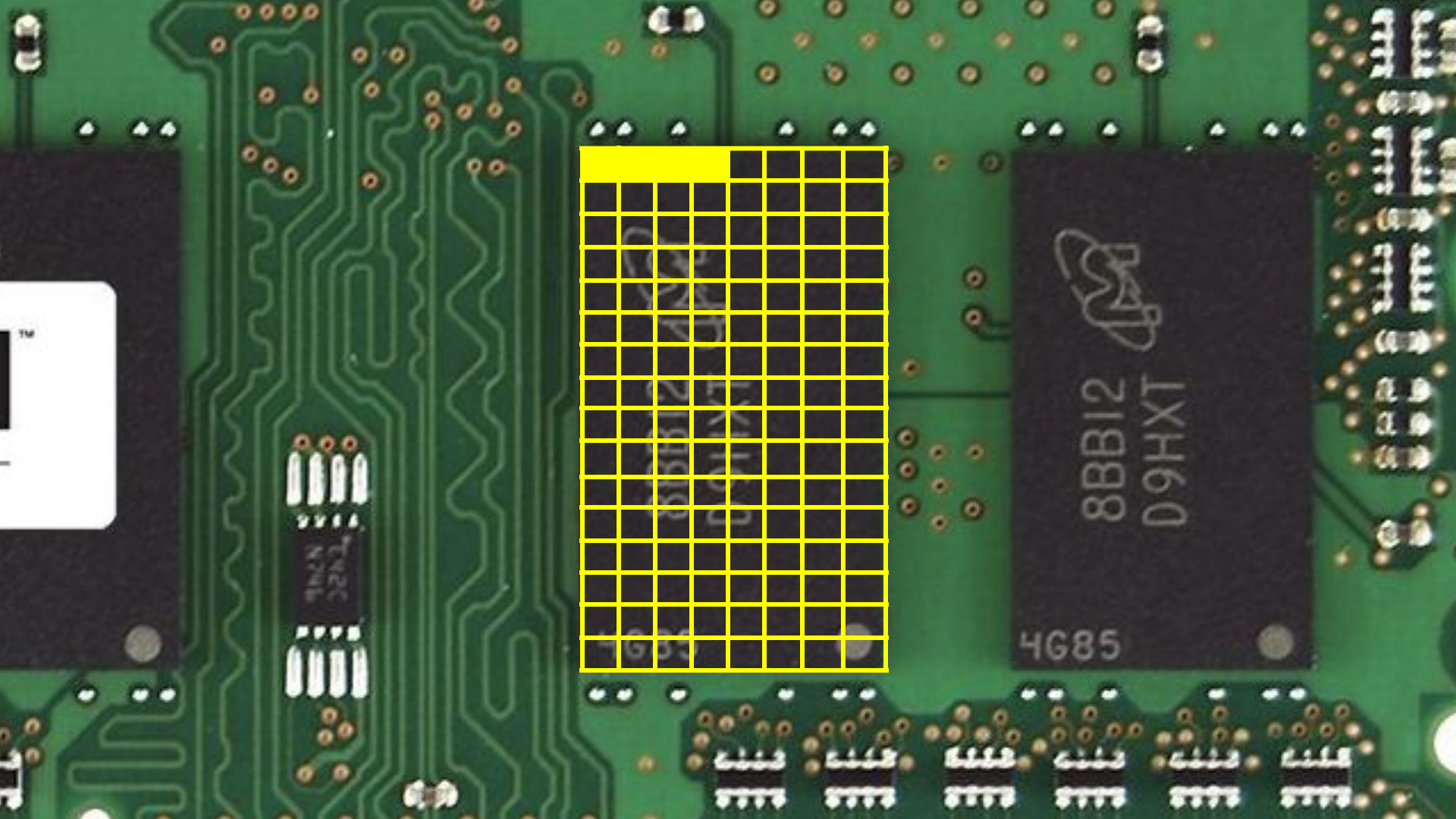

crucial
by Micron
8BBI2
D9HXT
4G85
8BBI2
D9HXT
4G85
4G85
47J6
4G85

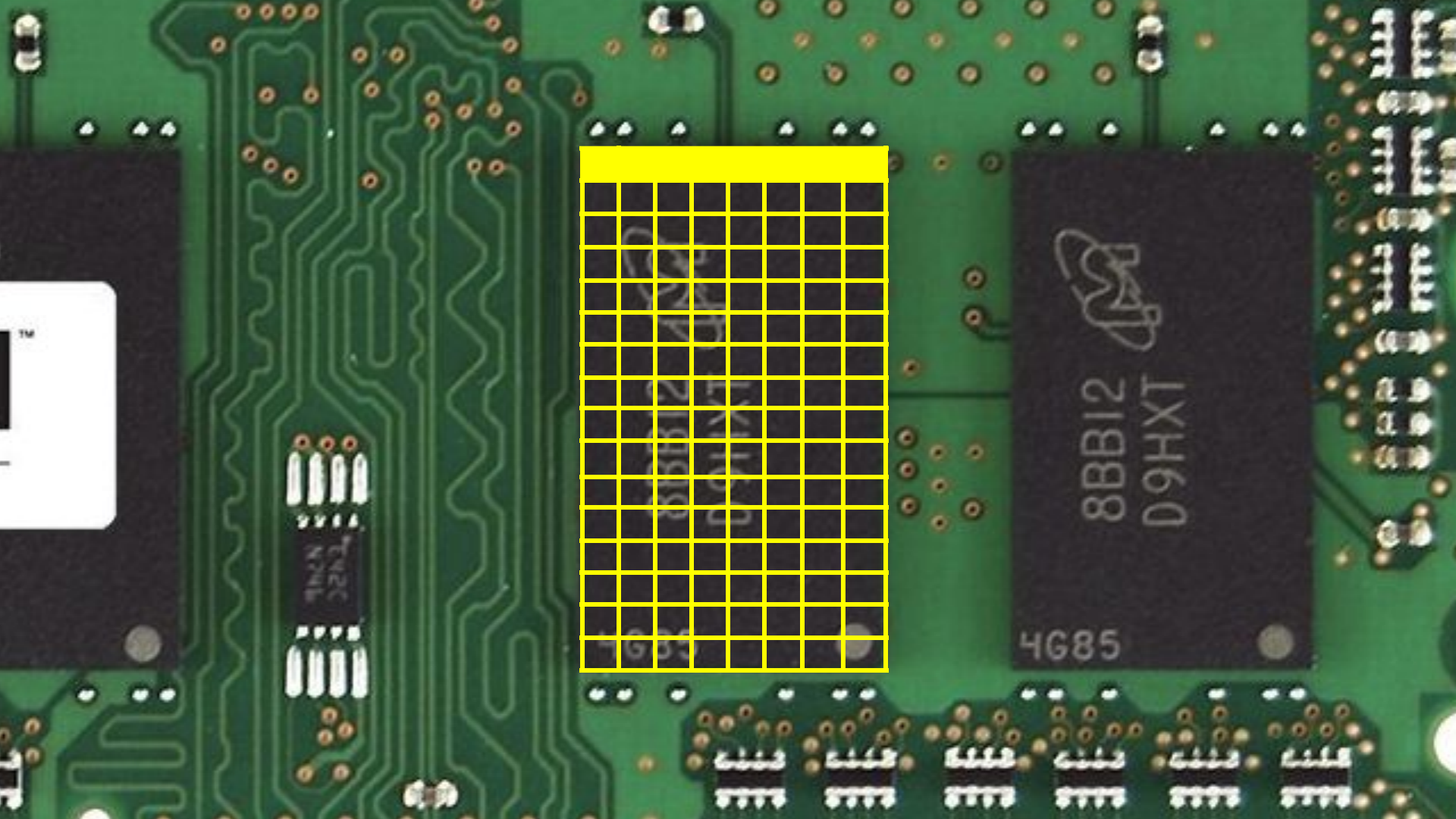

8BBI2
D9HXT
4G85
8BBI2
D9HXT
4G85

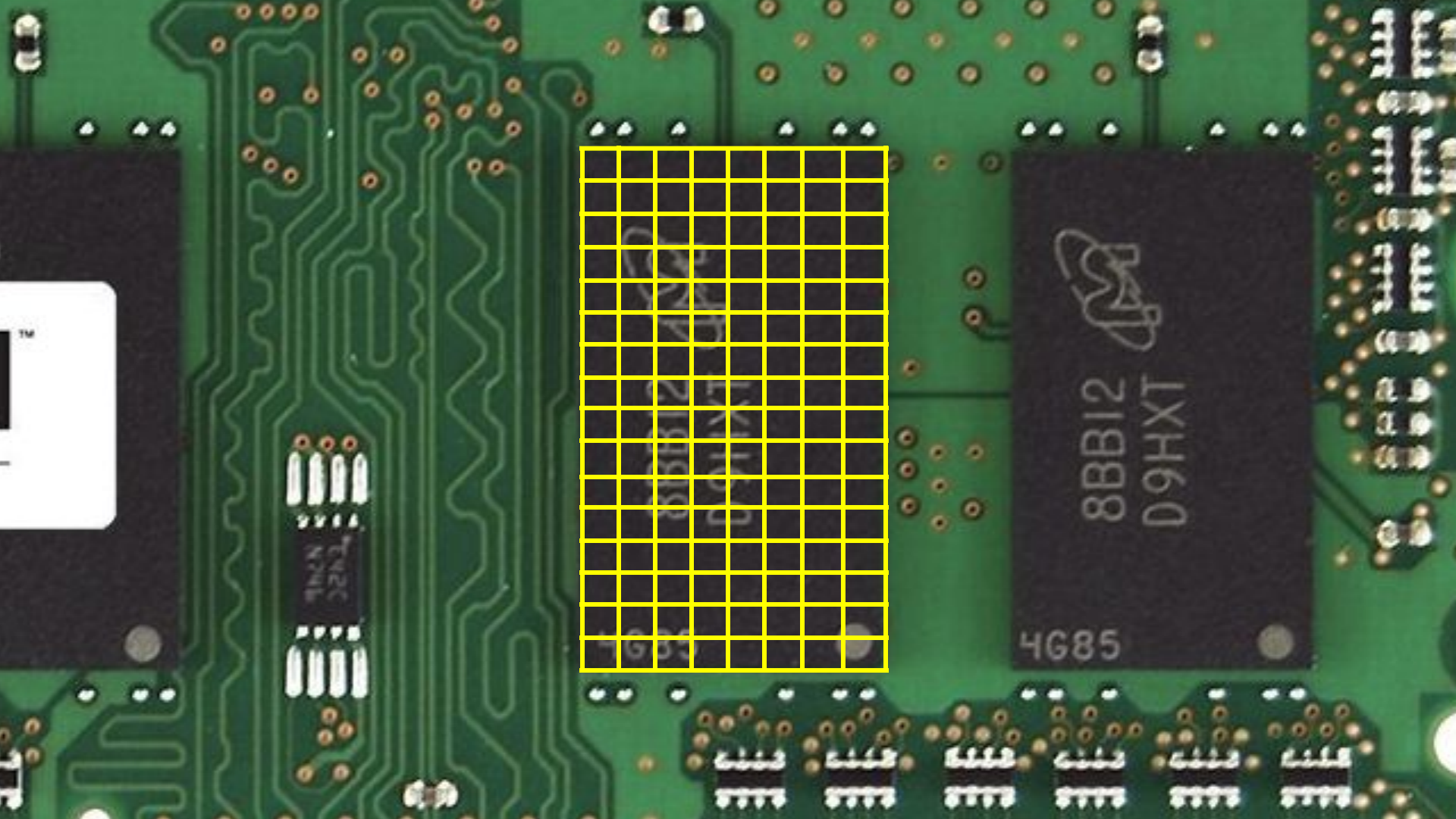

8BBI2
D9HXT
4G85
8BBI2
D9HXT
4G85

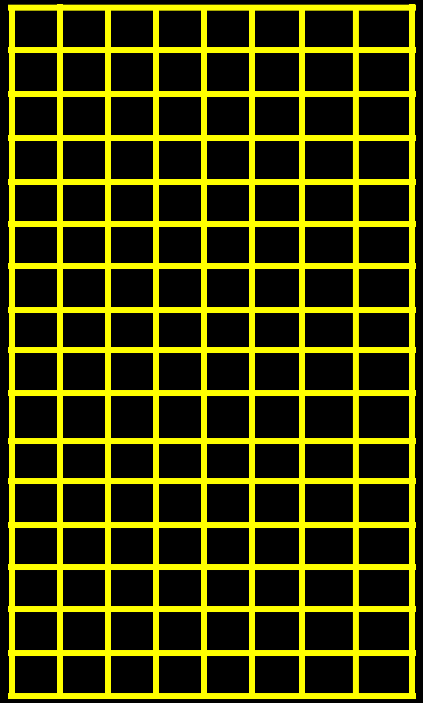

8BBI2
D9HXT
4G85


8BBI2
D9HXT
4G85

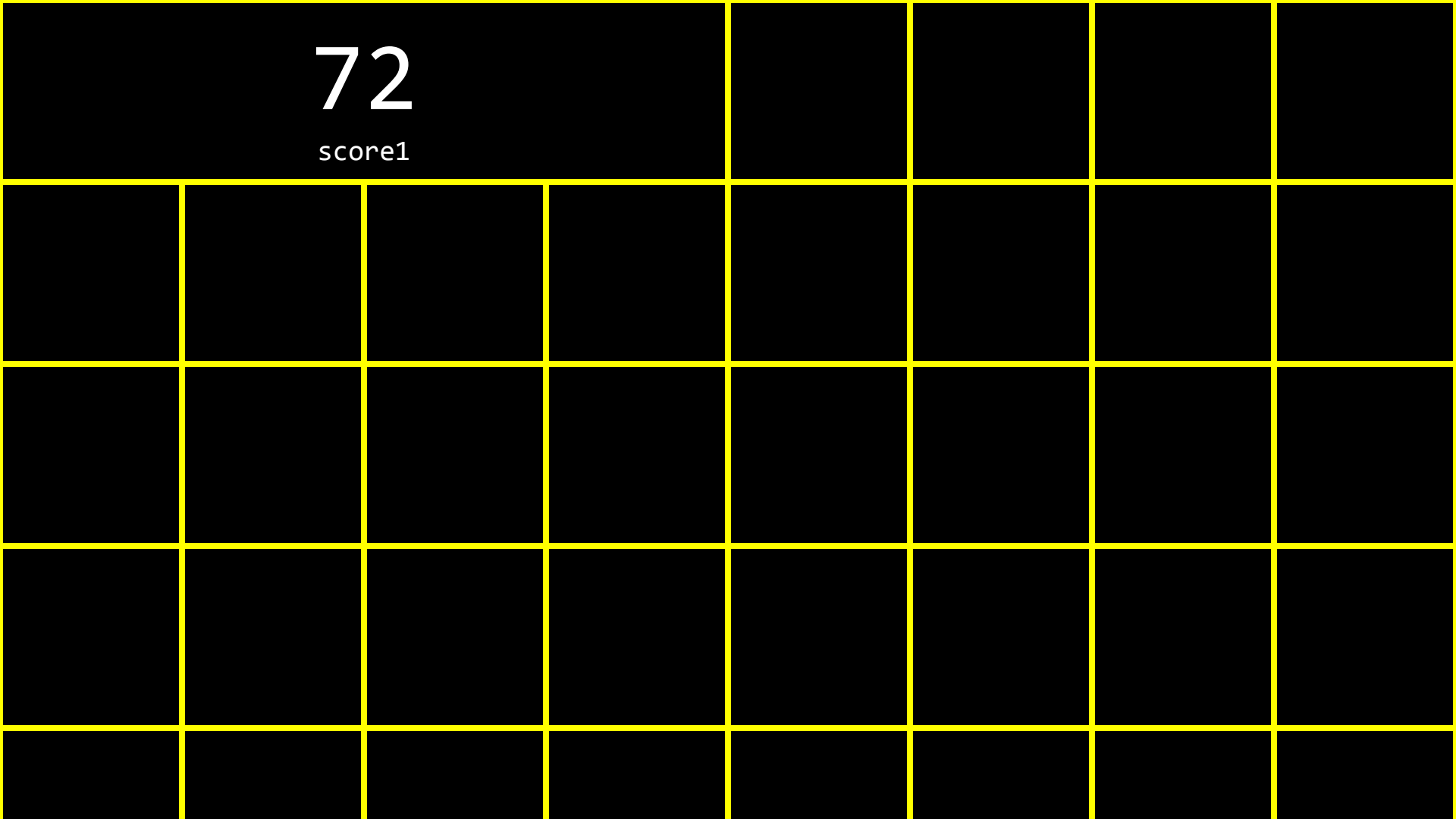

8BBI2
D9HXT
4G85


8BBI2
D9HXT
4G85


8BBI2
D9HXT
4G85



```
int score1 = 72;

int score2 = 73;

int score3 = 33;
```


72
score1

72

score1

73

score2

72

score1

73

score2

33

score3

00000000000000000000001001000

score1

00000000000000000000001001001

score2

00000000000000000000100001

score3

```
int score1 = 72;

int score2 = 73;

int score3 = 33;
```

arrays

```
int scores[3];
```

```
int scores[3];

scores[0] = 72;

scores[1] = 73;

scores[2] = 33;
```

72

scores[0]

73

scores[1]

33

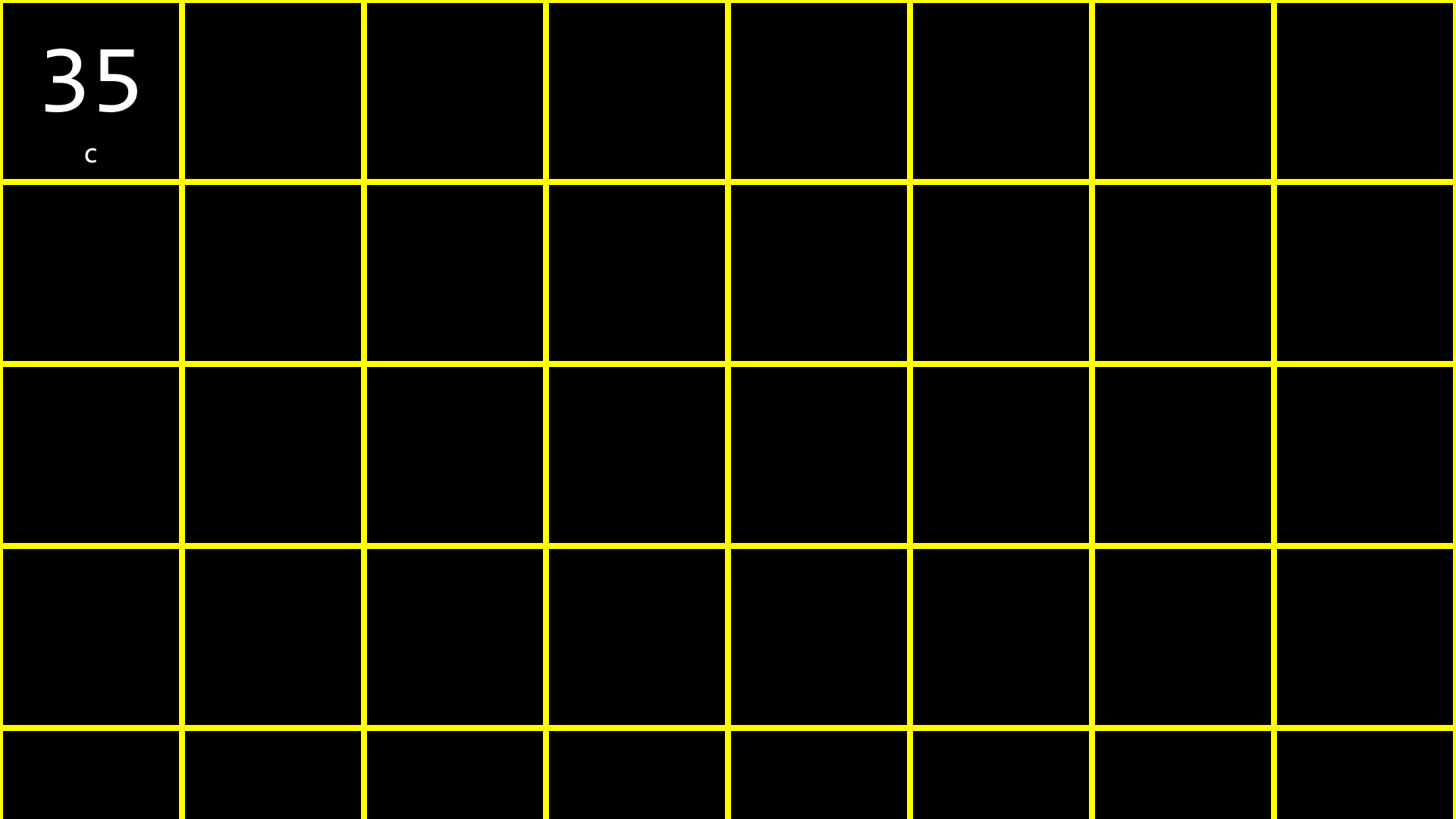
scores[2]

constants

```
char c = '#';
```

#

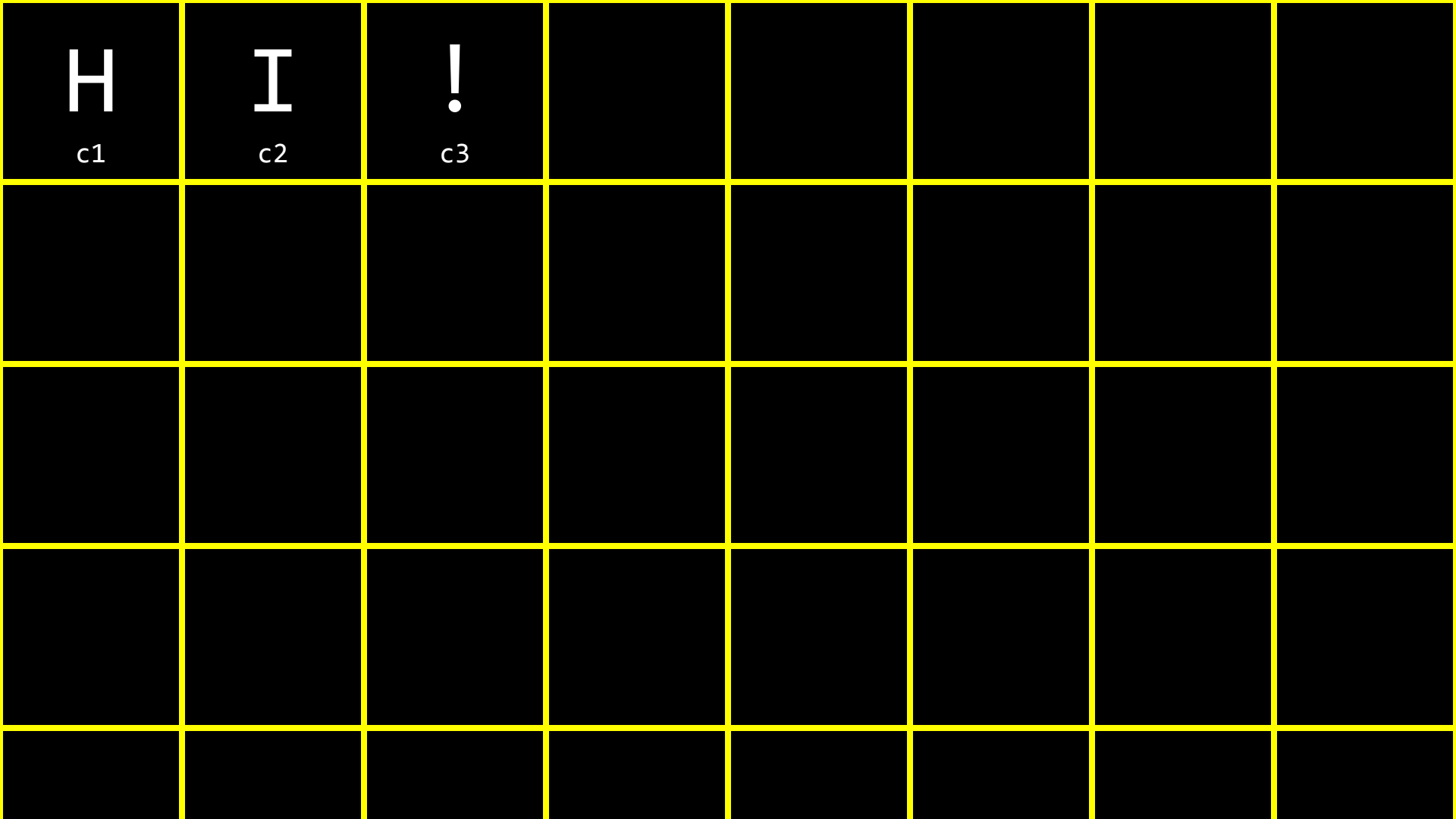
C


35
c

00100011

c

```
char c1 = 'H';

char c2 = 'I';

char c3 = '!';
```


H
I
!
c1
c2
c3

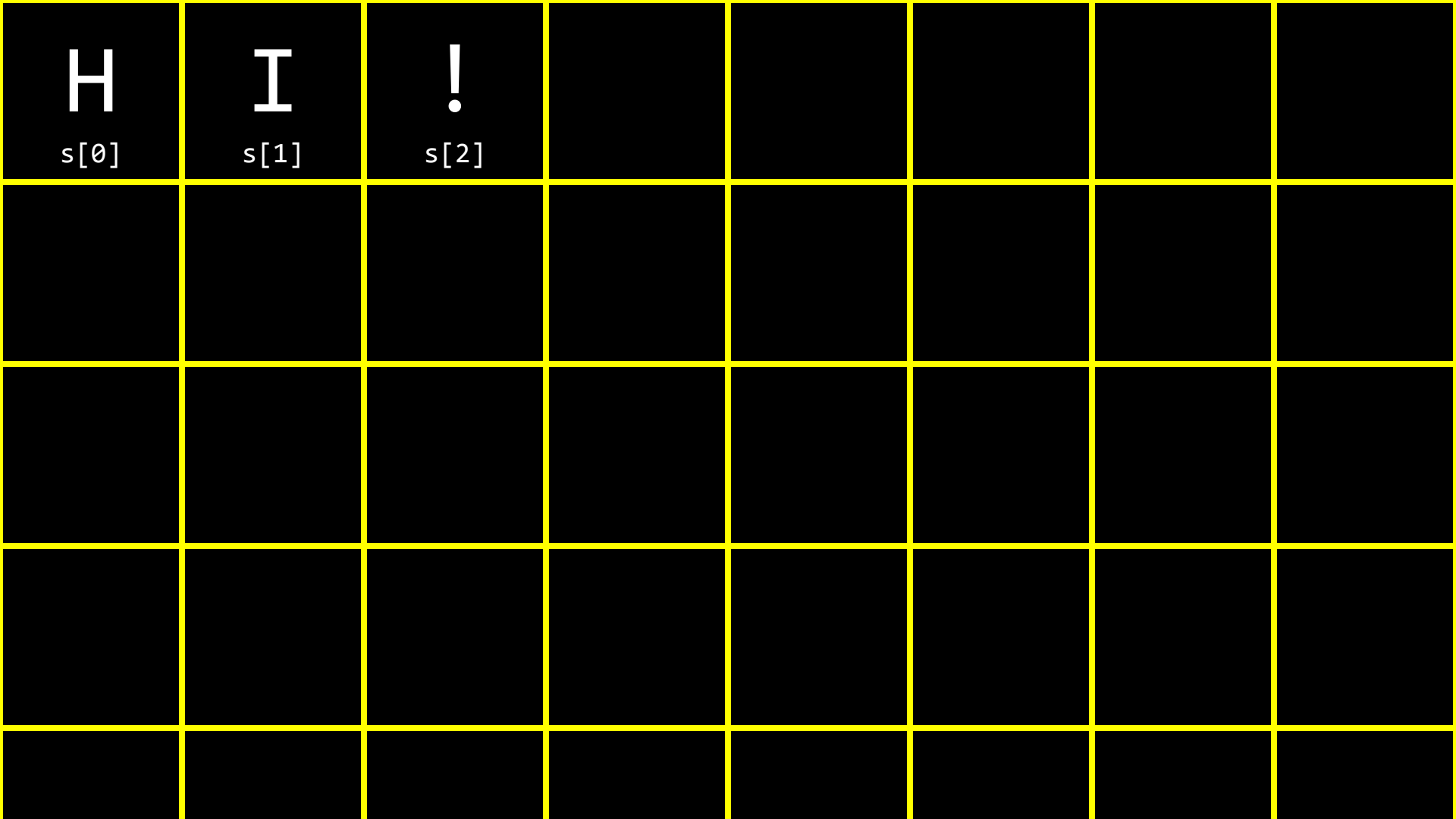
72 c1	73 c2	33 c3					

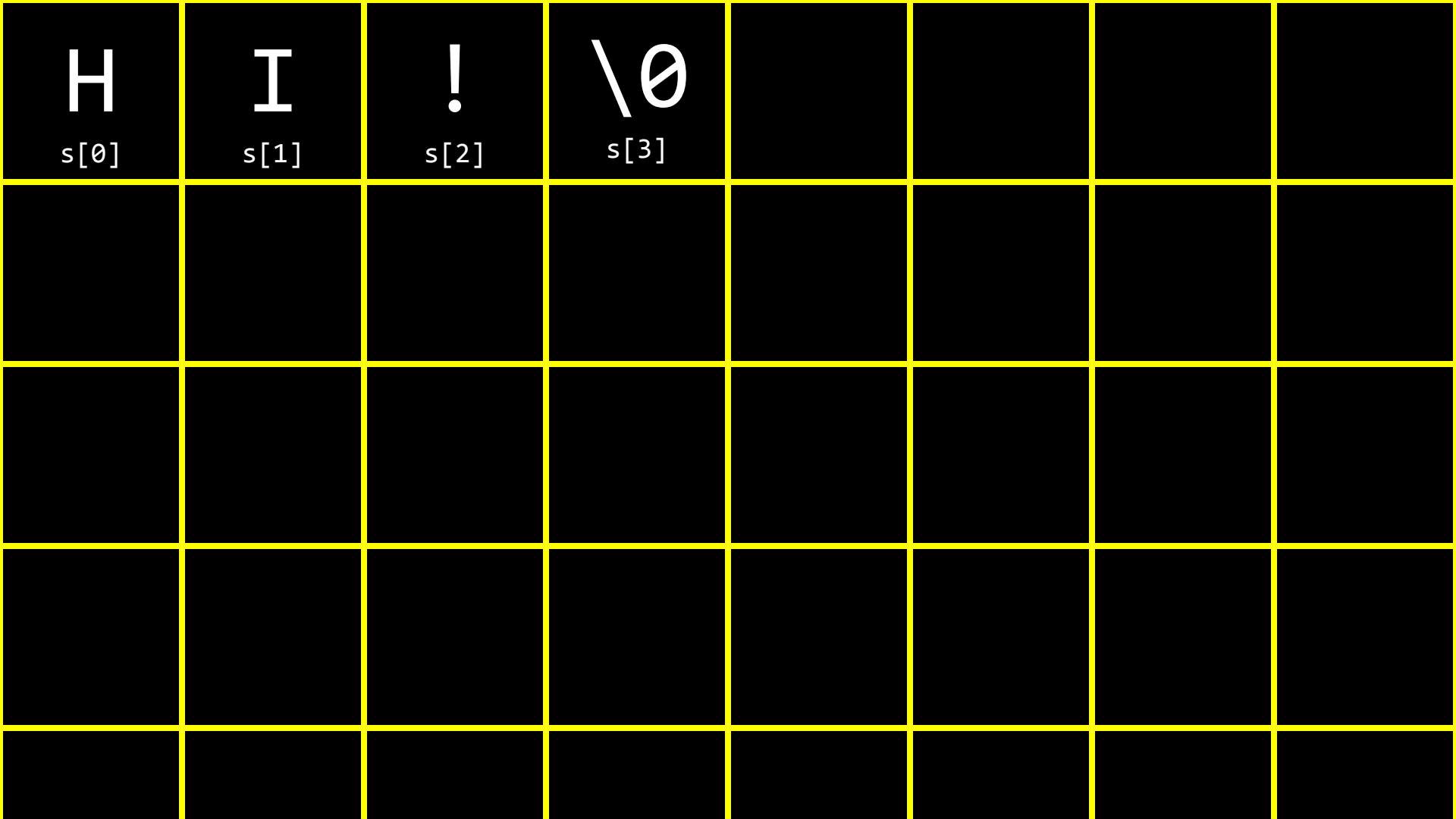
01001000 c1	01001001 c2	00100001 c3					

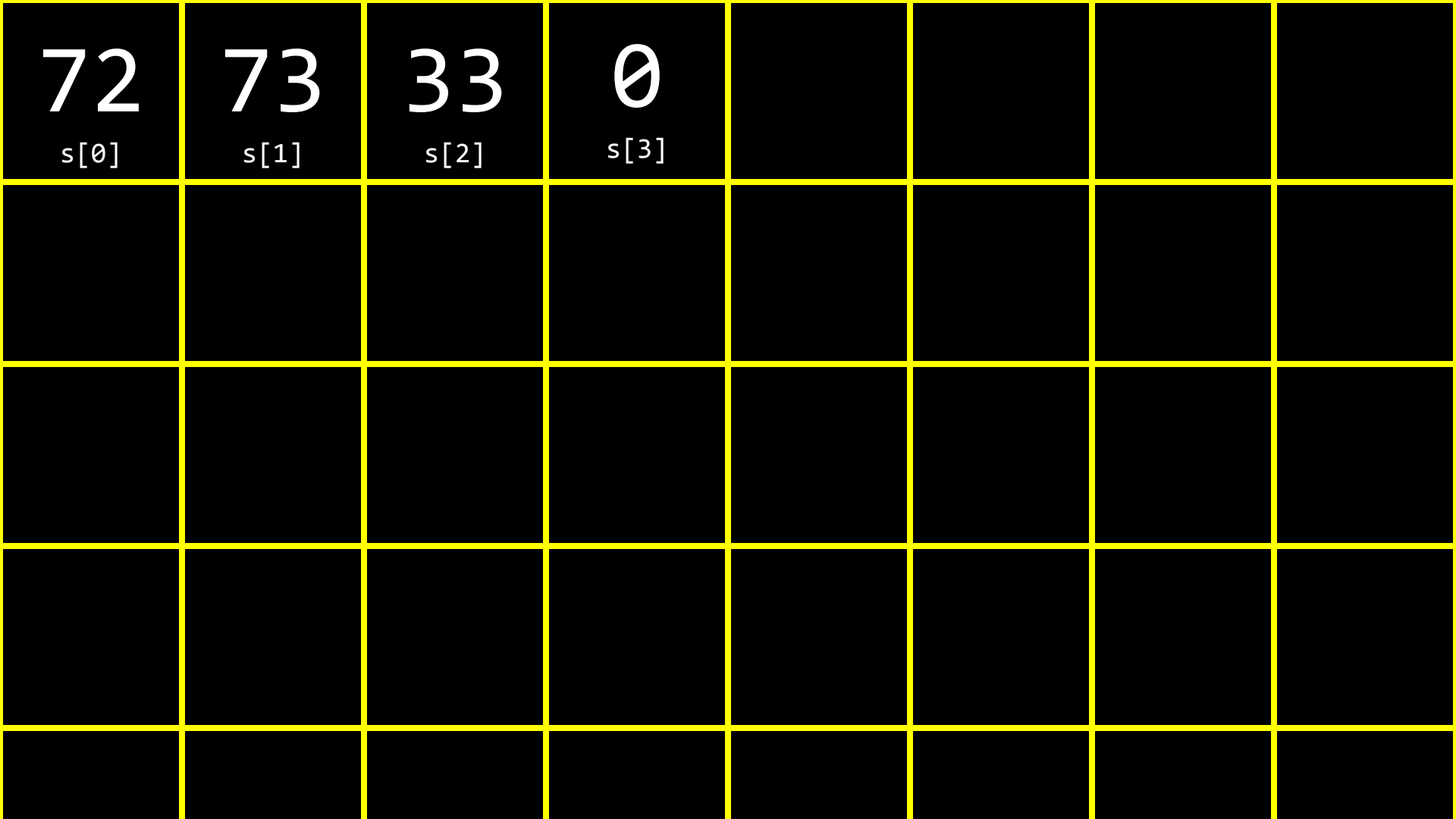
```
string s = "HI!";
```

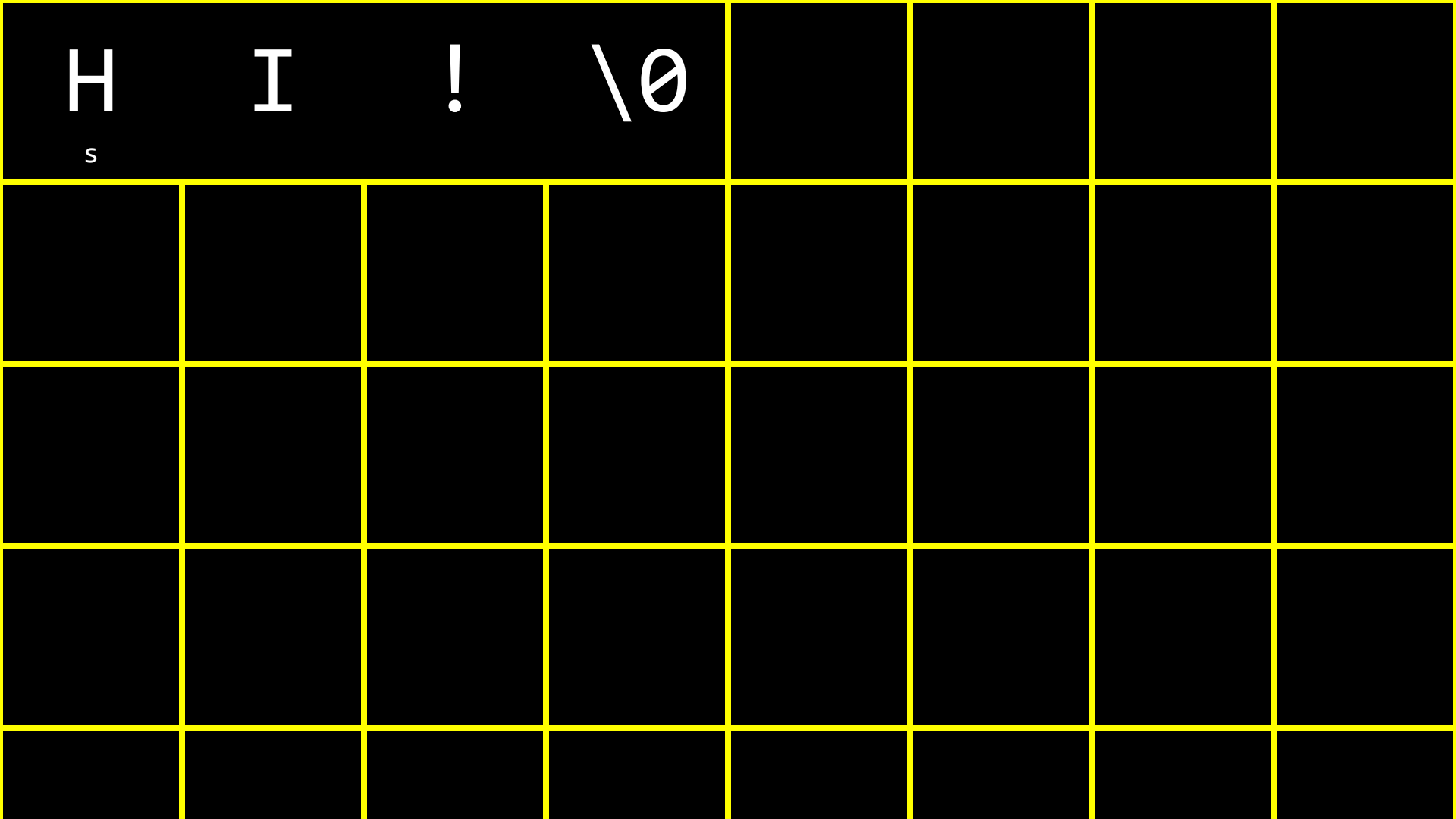
H I !

s


H
I
!
s[0]
s[1]
s[2]

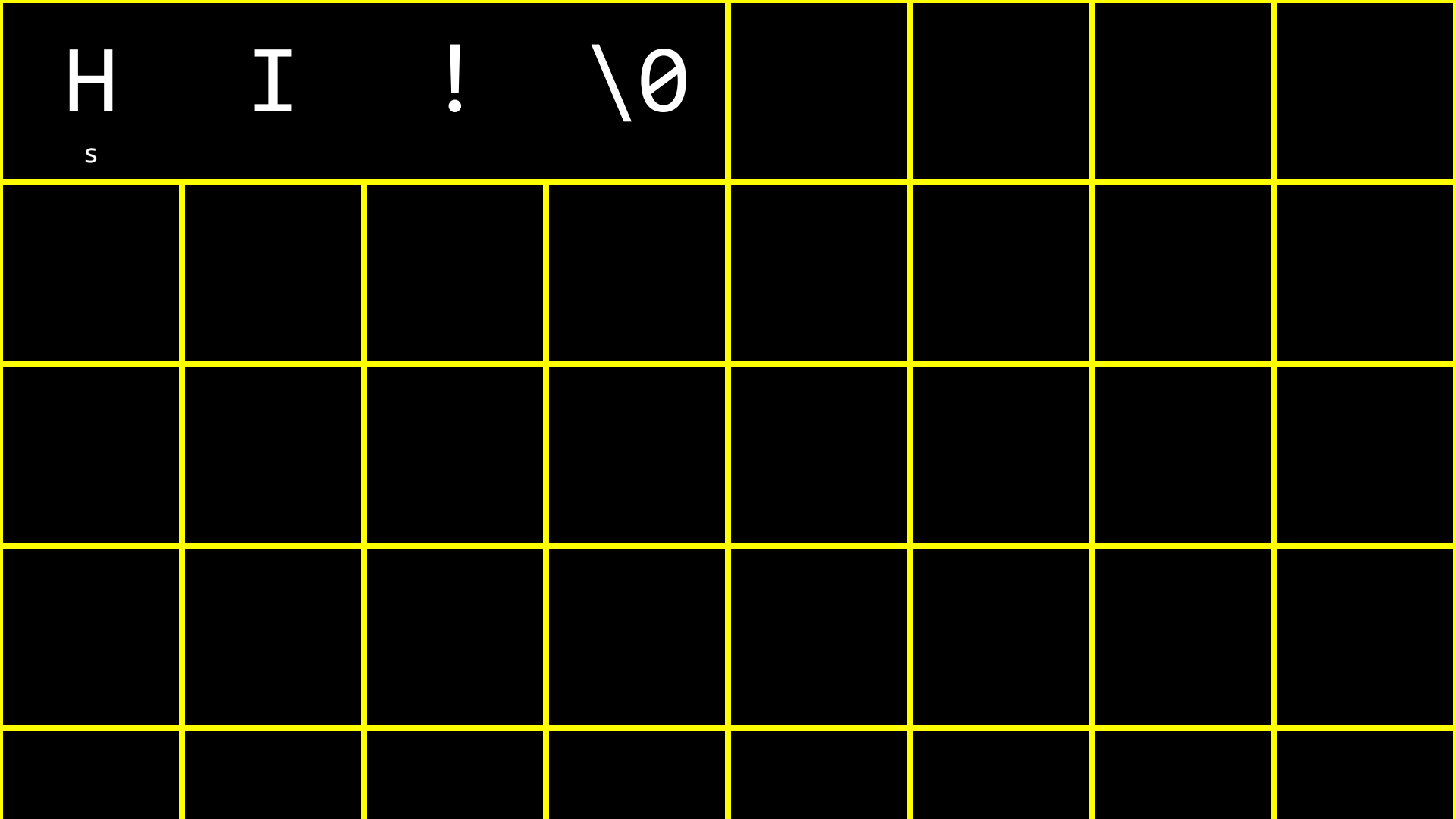

H
I
!
\0
s[0]
s[1]
s[2]
s[3]

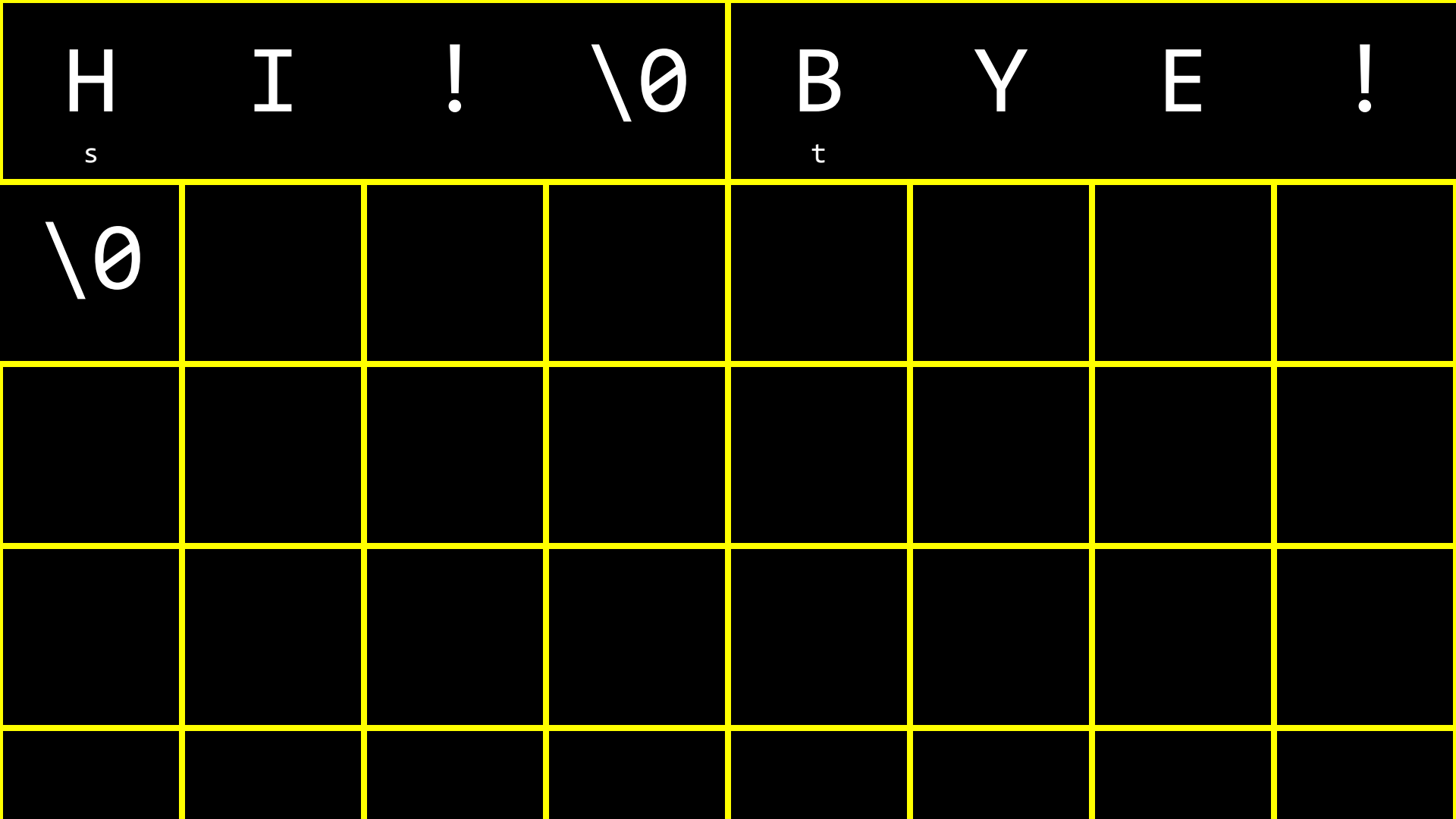

72
s[0]
73
s[1]
33
s[2]
0
s[3]

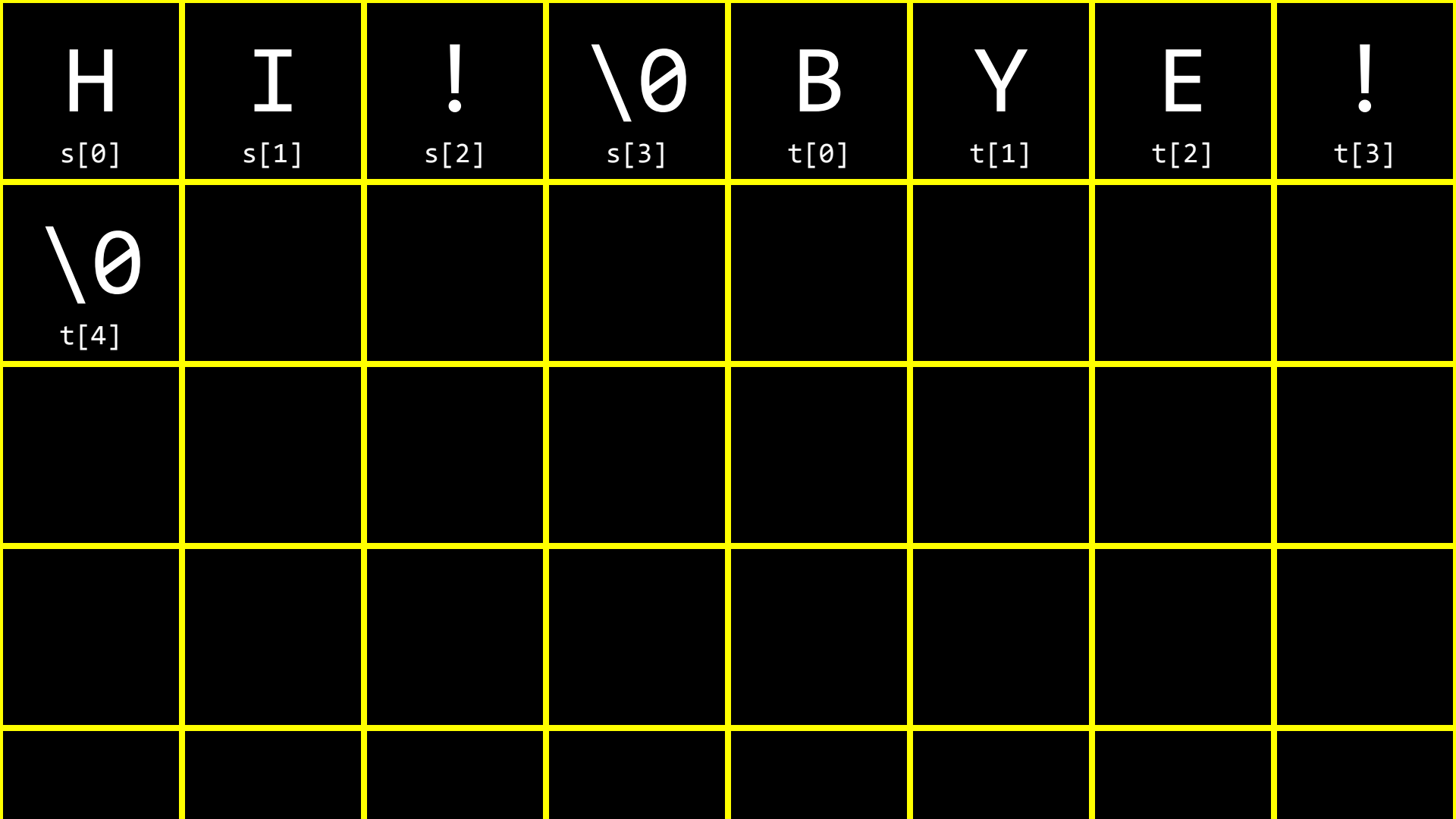

H I ! \0
s

NUL

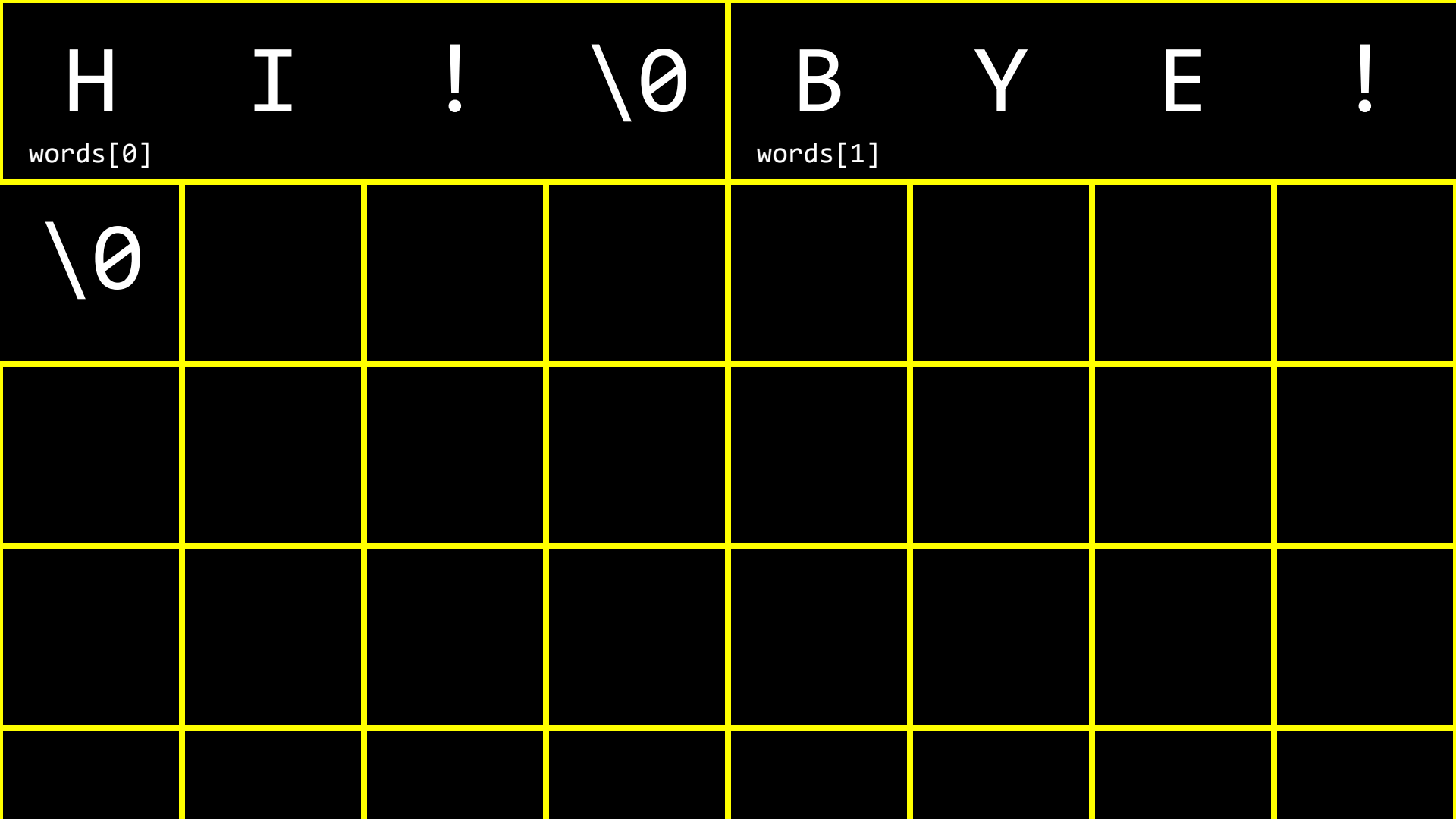
```
string s = "HI!";

string t = "BYE!";
```

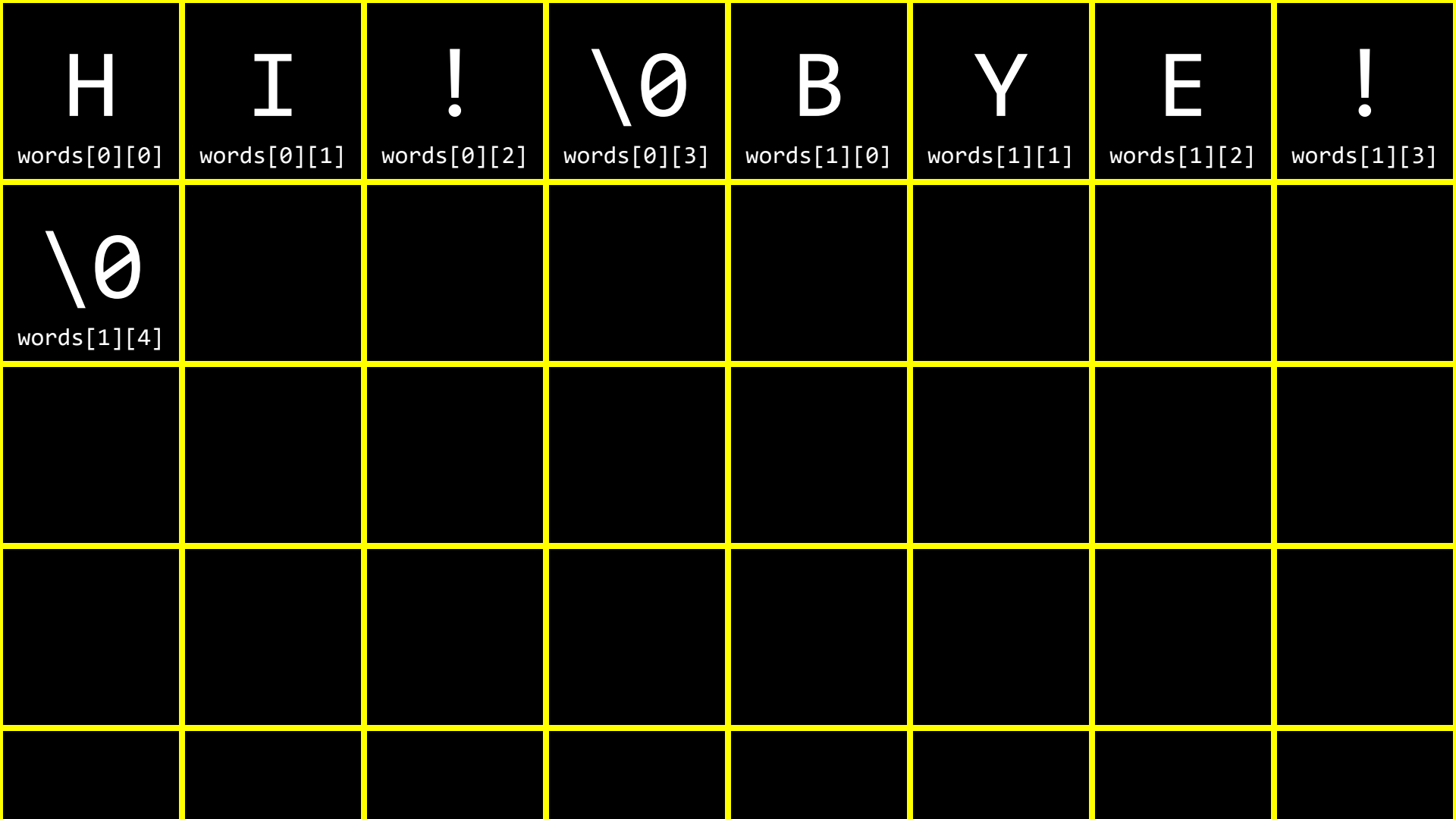

H I ! \0
s


H I ! \0
s
B Y E !
t
\0


H
s[0]
I
s[1]
!
s[2]
\0
s[3]
B
t[0]
Y
t[1]
E
t[2]
!
t[3]
\0
t[4]

```
string words[2];

words[0] = "HI!";

words[1] = "BYE!";
```


H I ! \0
words[0]
B Y E !
words[1]
\0


H
words[0][0]
I
words[0][1]
!
words[0][2]
\0
words[0][3]
B
words[1][0]
Y
words[1][1]
E
words[1][2]
!
words[1][3]
\0
words[1][4]

string

manual pages

command-line arguments

```c
#include <stdio.h>

int main(void)
{
    ...
}
```

```c
#include <stdio.h>

int main(void)
{
    ...
}
```

```
#include <stdio.h>

int main(int argc, string argv[])
{
    ...
}
```

exit status

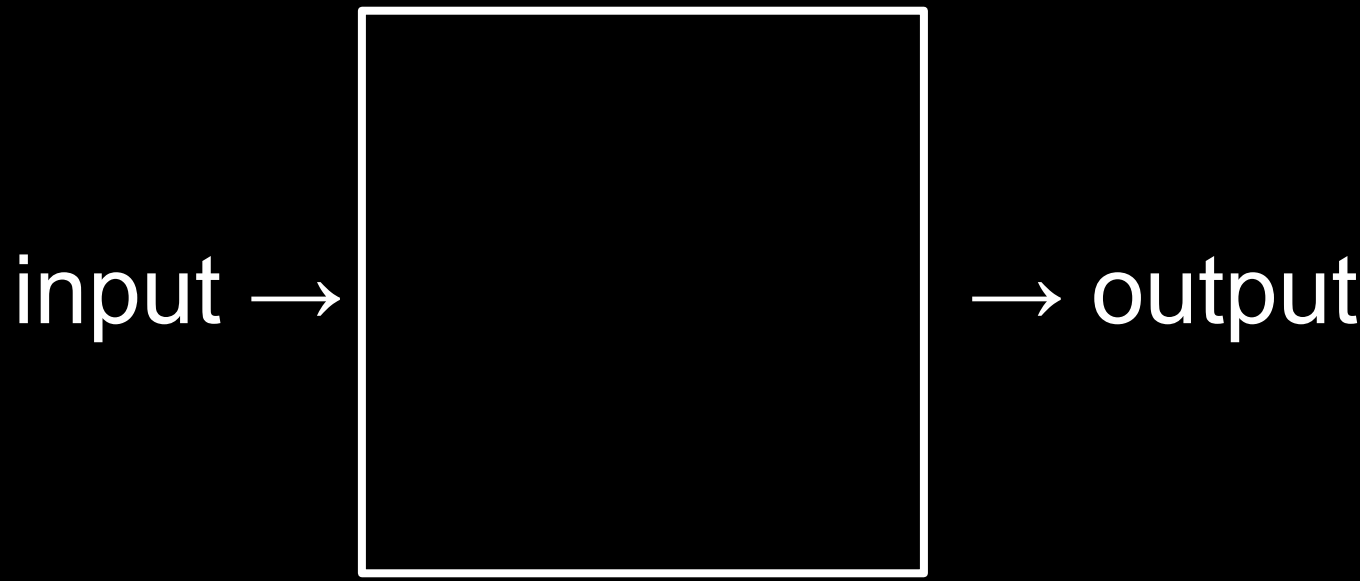
```
#include <stdio.h>

int main(int argc, string argv[])
{
    ...
}
```

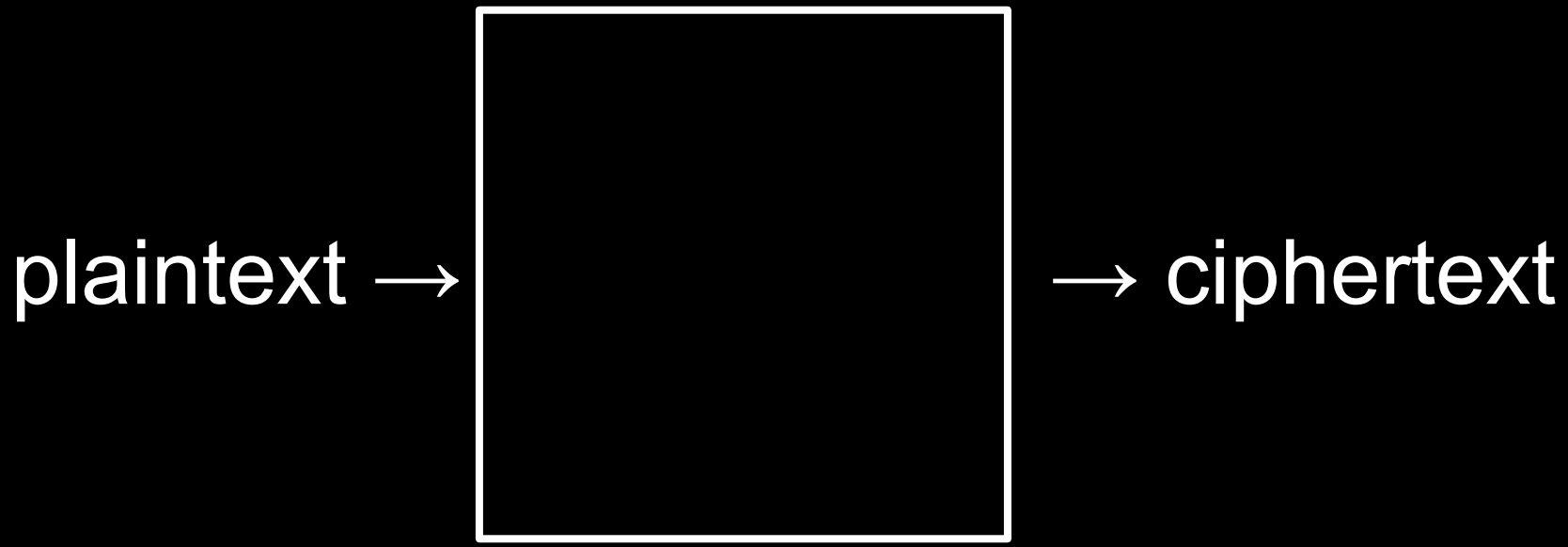
```
#include <stdio.h>

int main(int argc, string argv[])
{
    ...
}
```

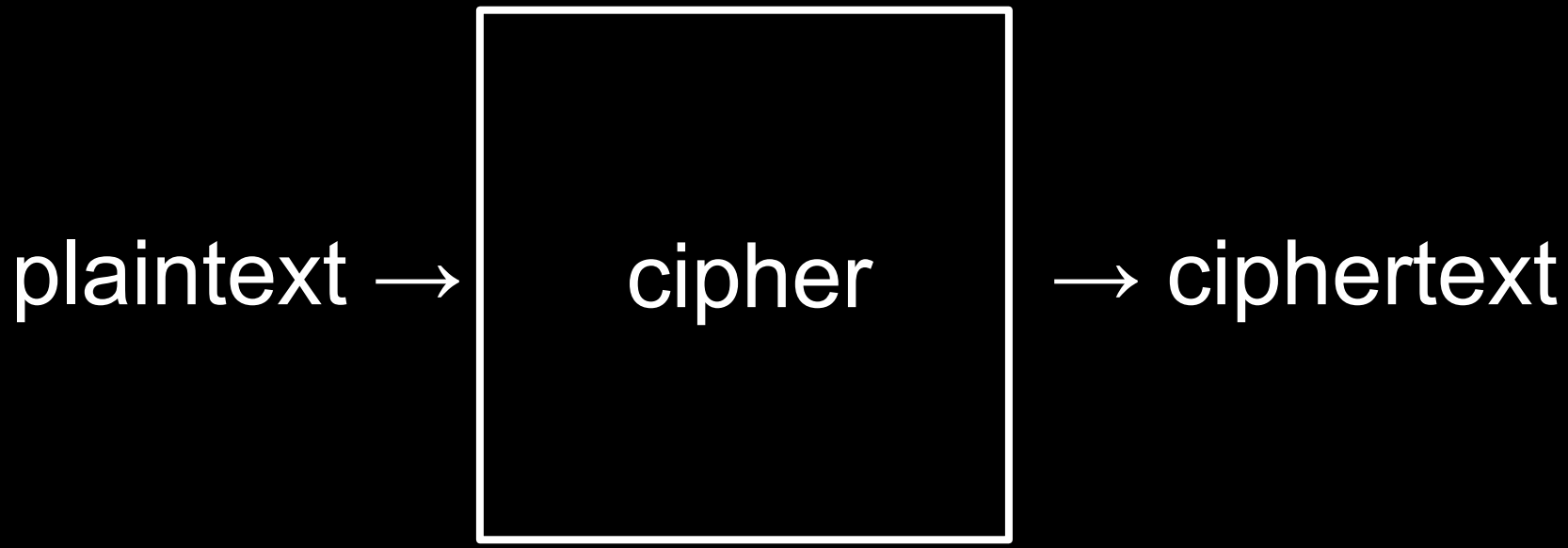
```c
#include <stdio.h>

int main(void)
{
    ...
}
```

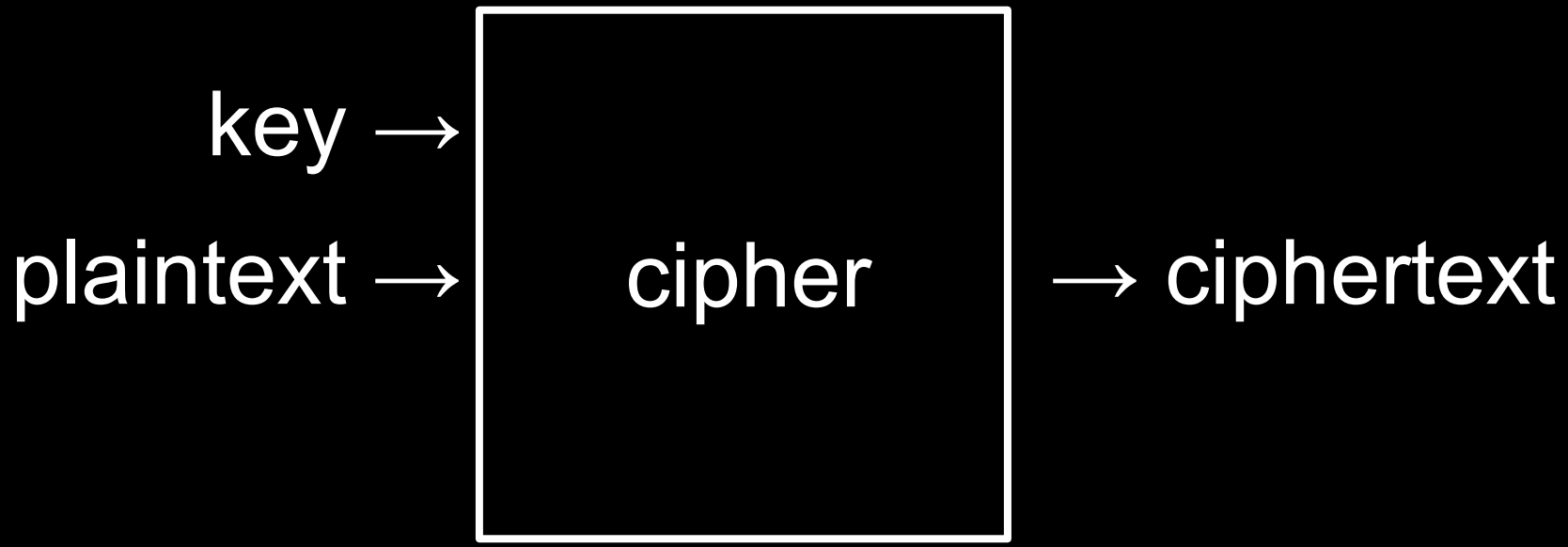
readability

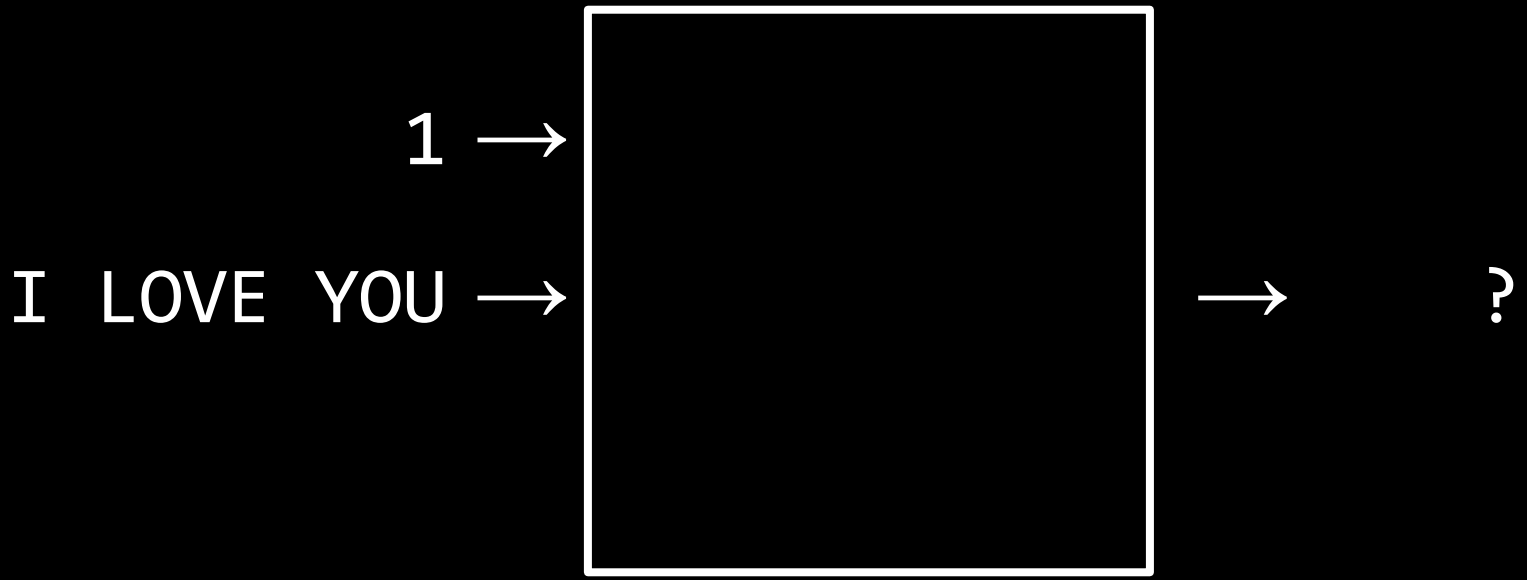
cryptography


input →
→ output


plaintext →
→ ciphertext


plaintext →
cipher
→ ciphertext


key →
plaintext →
cipher
→ ciphertext


1
I LOVE YOU
?

I L O V E Y O U

73 L O V E Y O U

73 76 O V E Y O U

73 76 79 V E Y O U

73 76 79 86 E Y O U

73 76 79 86 69 Y O U

73 76 79 86 69 89 O U

73 76 79 86 69 89 79 U

73 76 79 86 69 89 79 85

74 76 79 86 69 89 79 85

74 77 79 86 69 89 79 85

74 77 80 86 69 89 79 85

74 77 80 87 69 89 79 85

74 77 80 87 70 89 79 85

74 77 80 87 70 90 79 85

74 77 80 87 70 90 80 85

74 77 80 87 70 90 80 86

```
J   77 80 87 70  90 80 86
```

J M 80 87 70 90 80 86

J M P 87 70 90 80 86

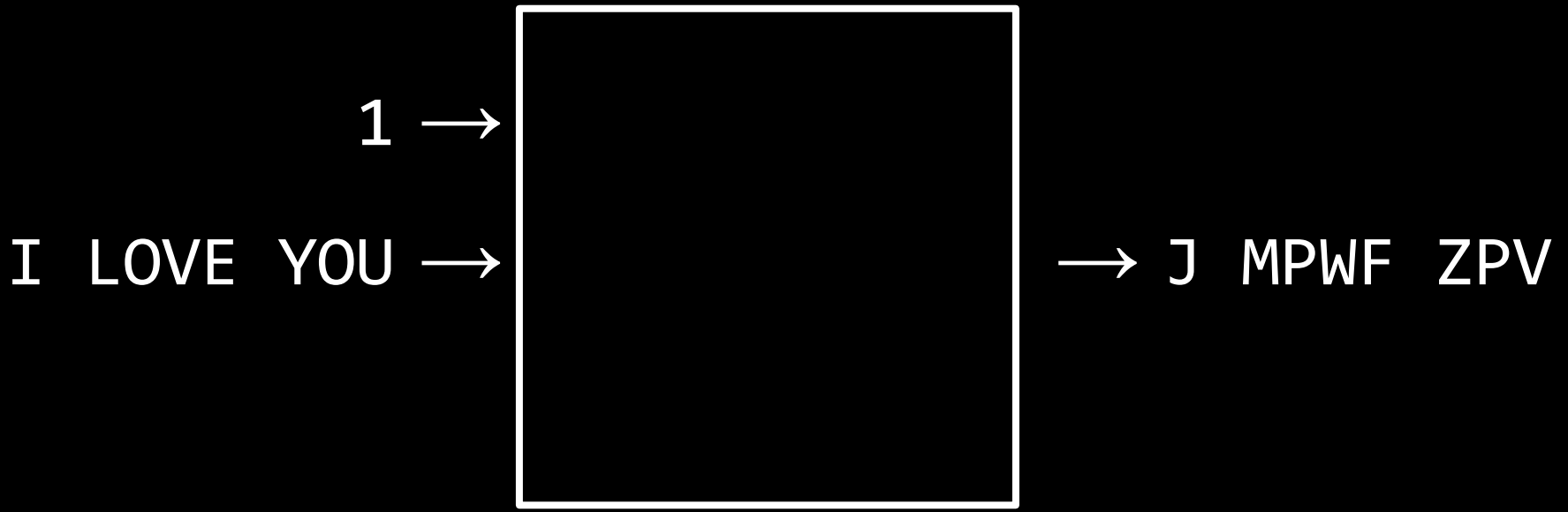
J M P W 70 90 80 86

J M P W F 90 80 86

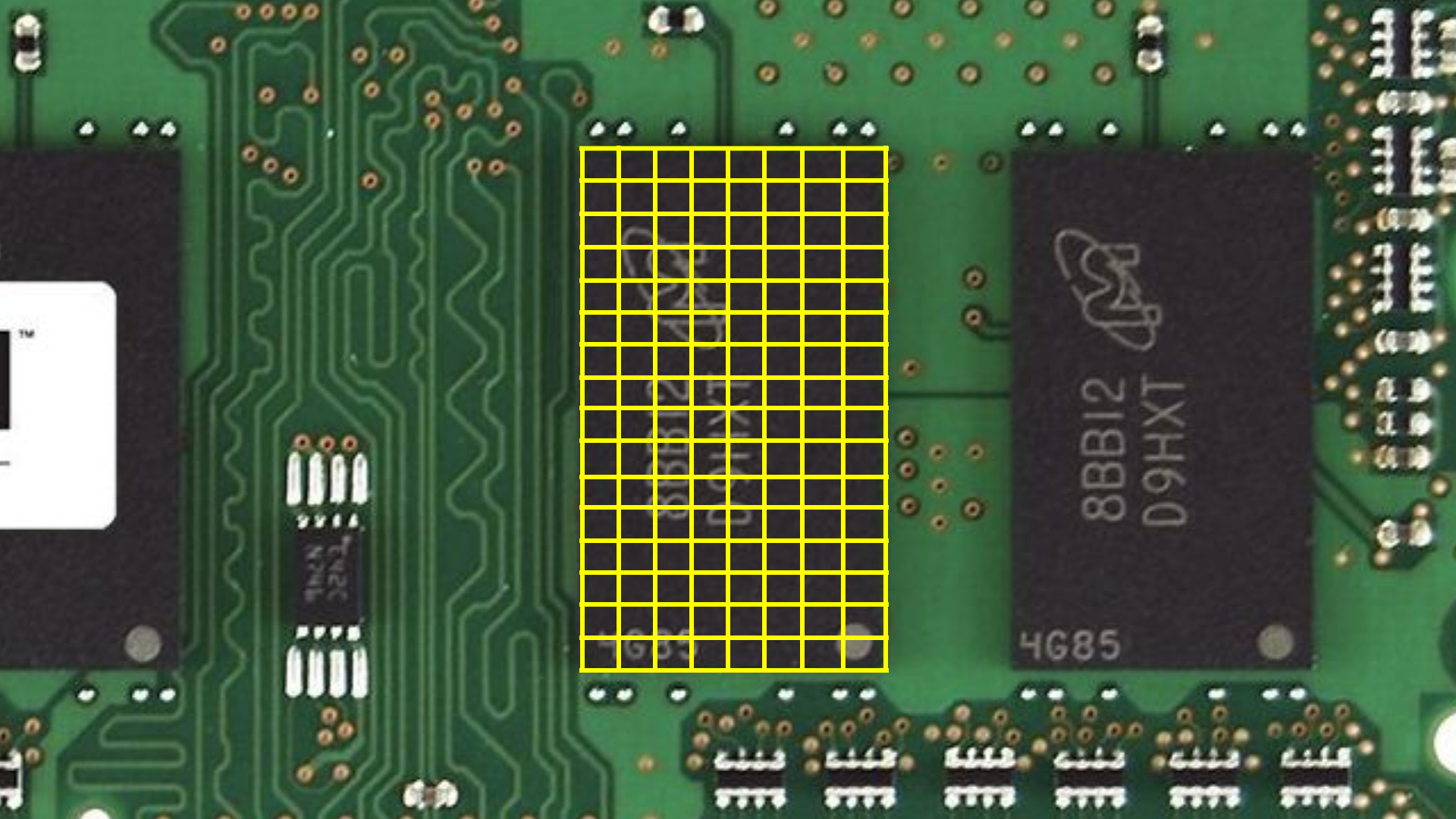
J M P W F Z 80 86

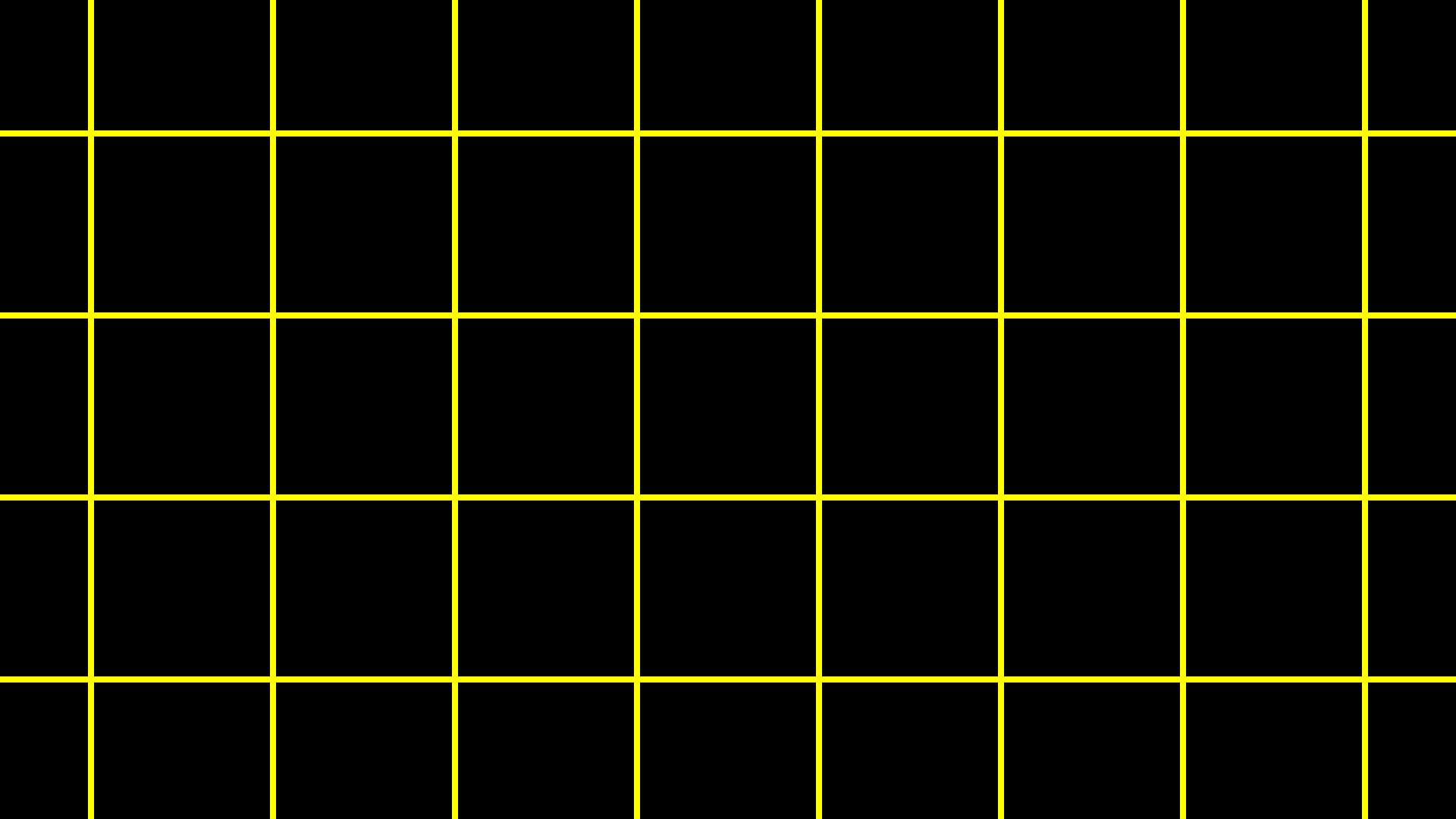
J M P W F Z P 86

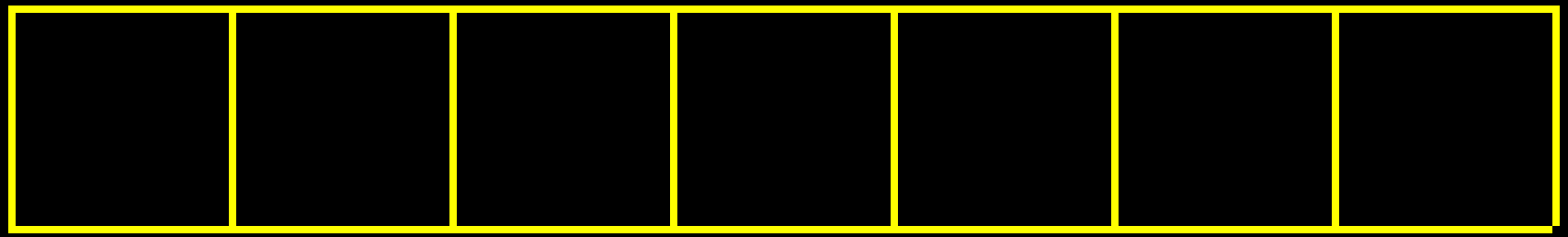
J M P W F Z P V

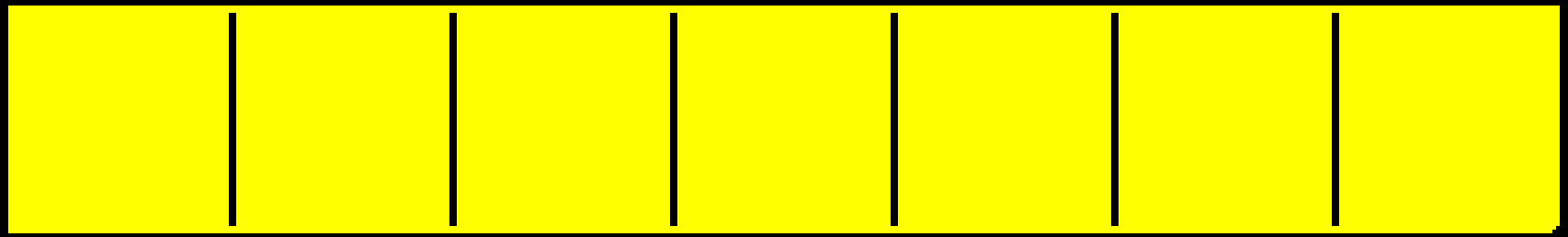

1 ⟶
I LOVE YOU ⟶
⟶ J MPWF ZPV


8BBI2
D9HXT
4G85
8BBI2
D9HXT
4G85

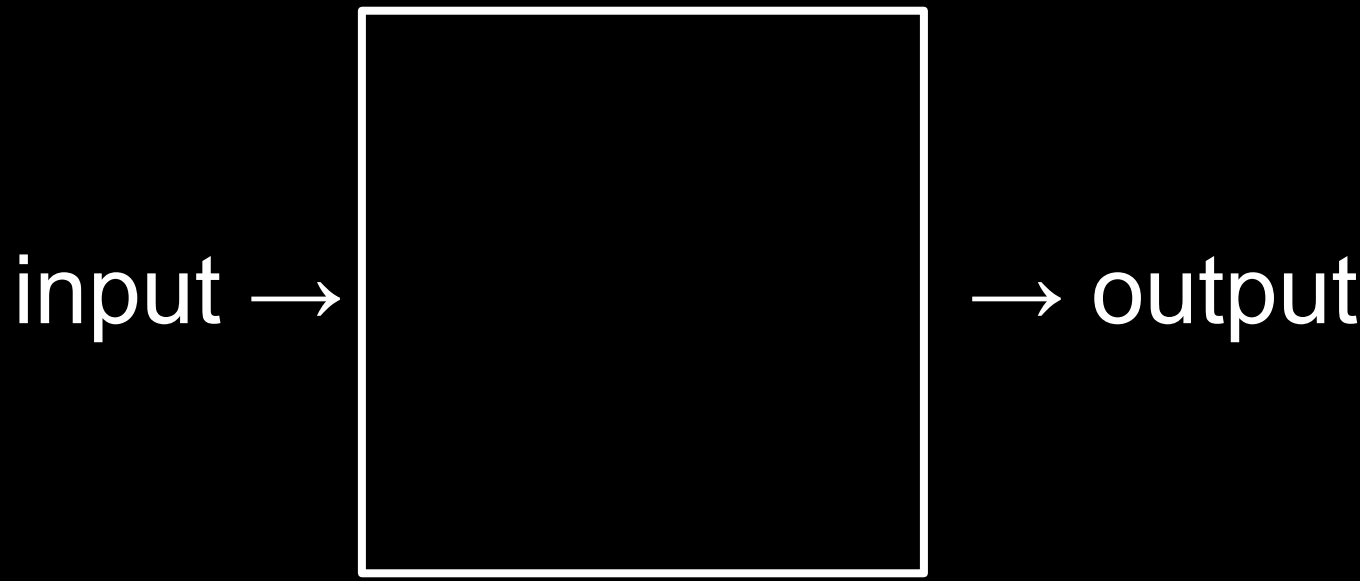

8BBI2
D9HXT
4G85

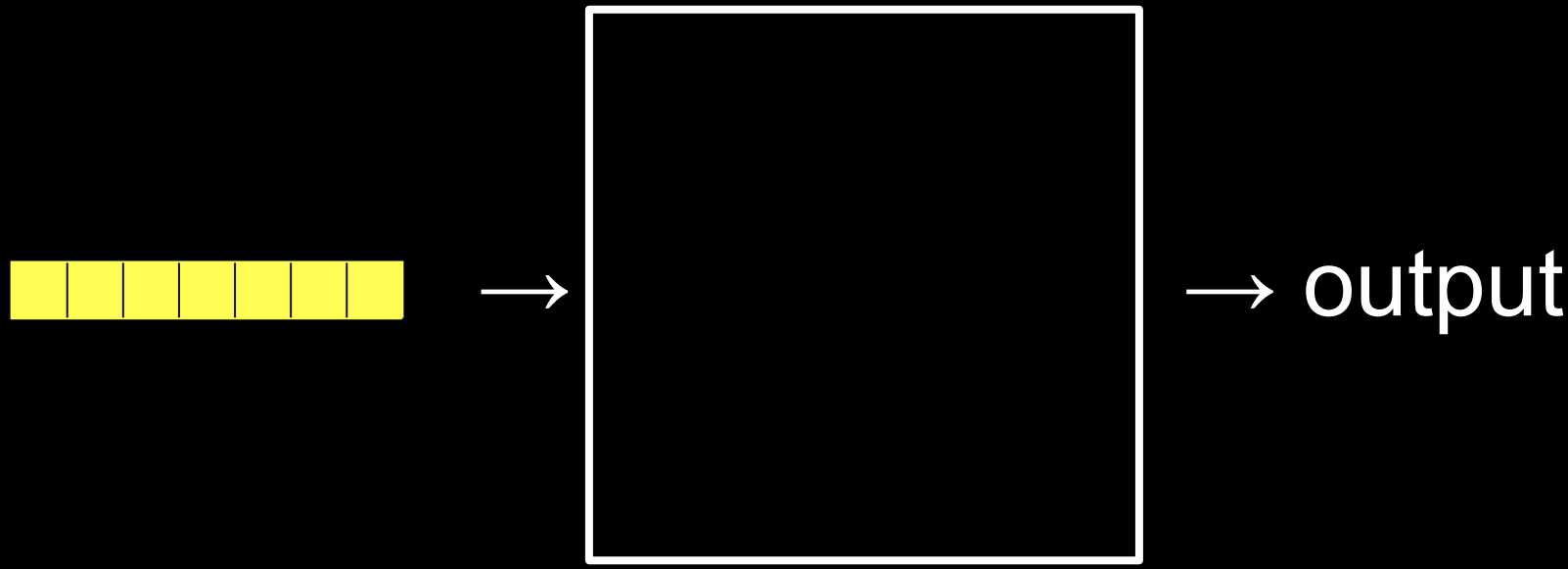


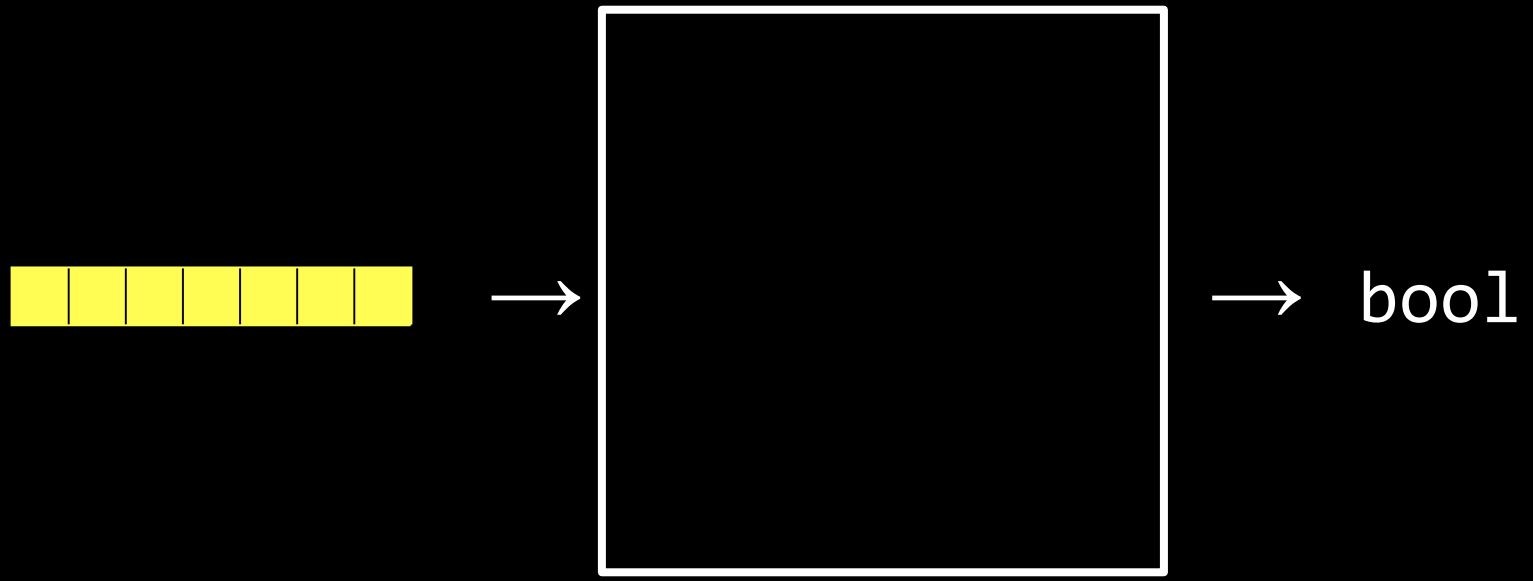


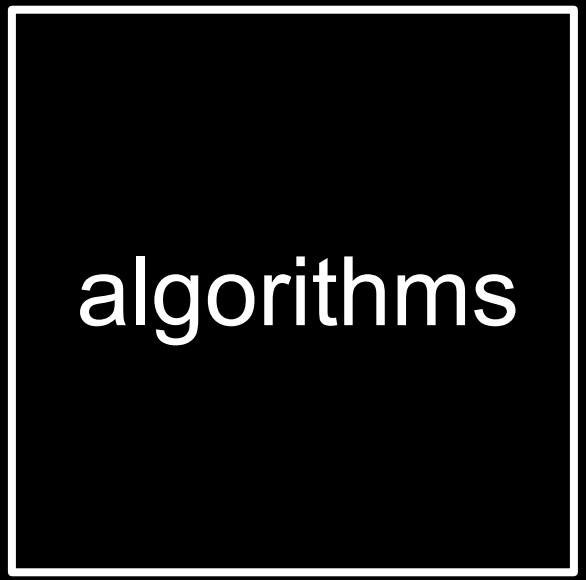


searching


input →
→ output

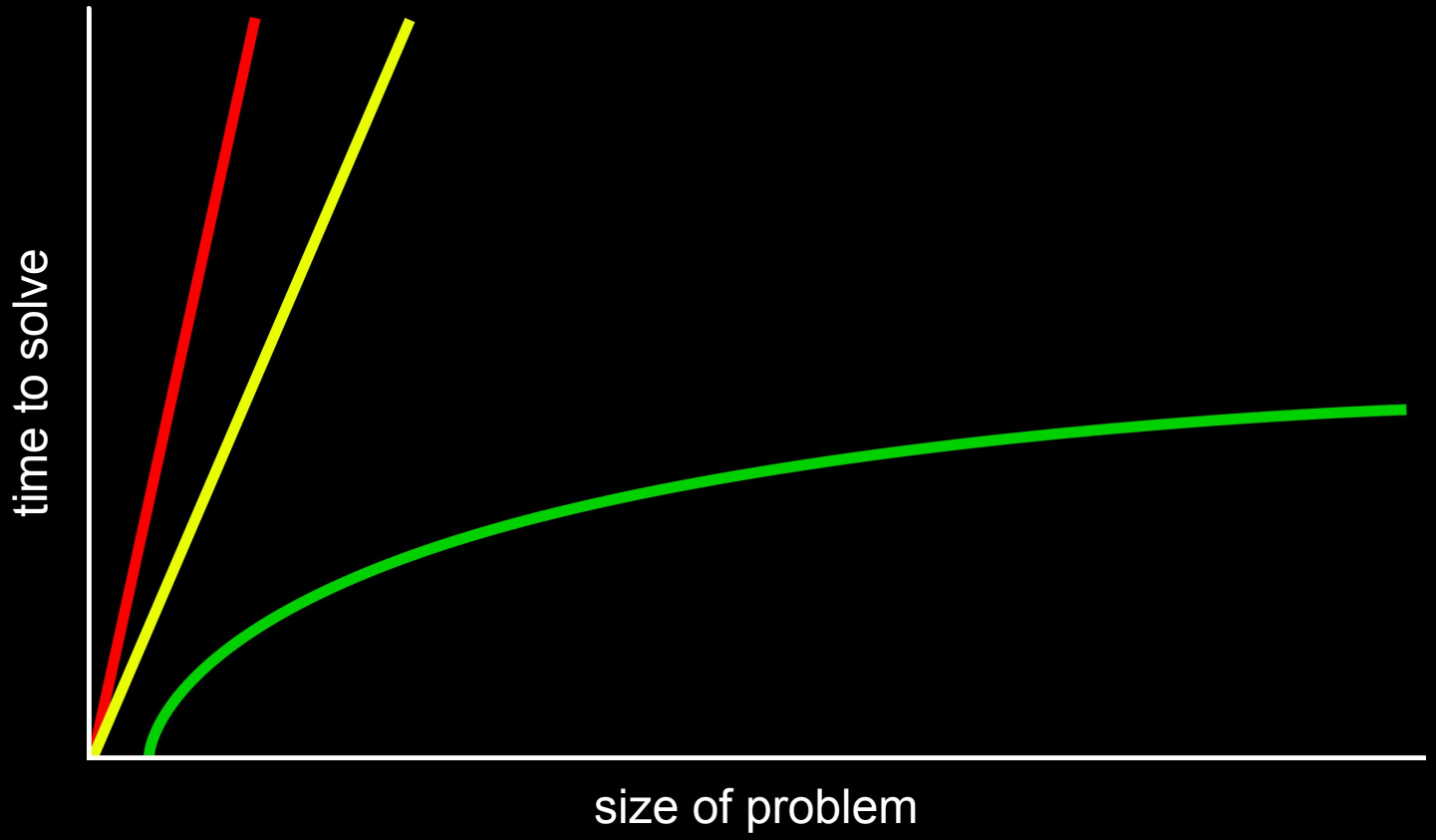

output

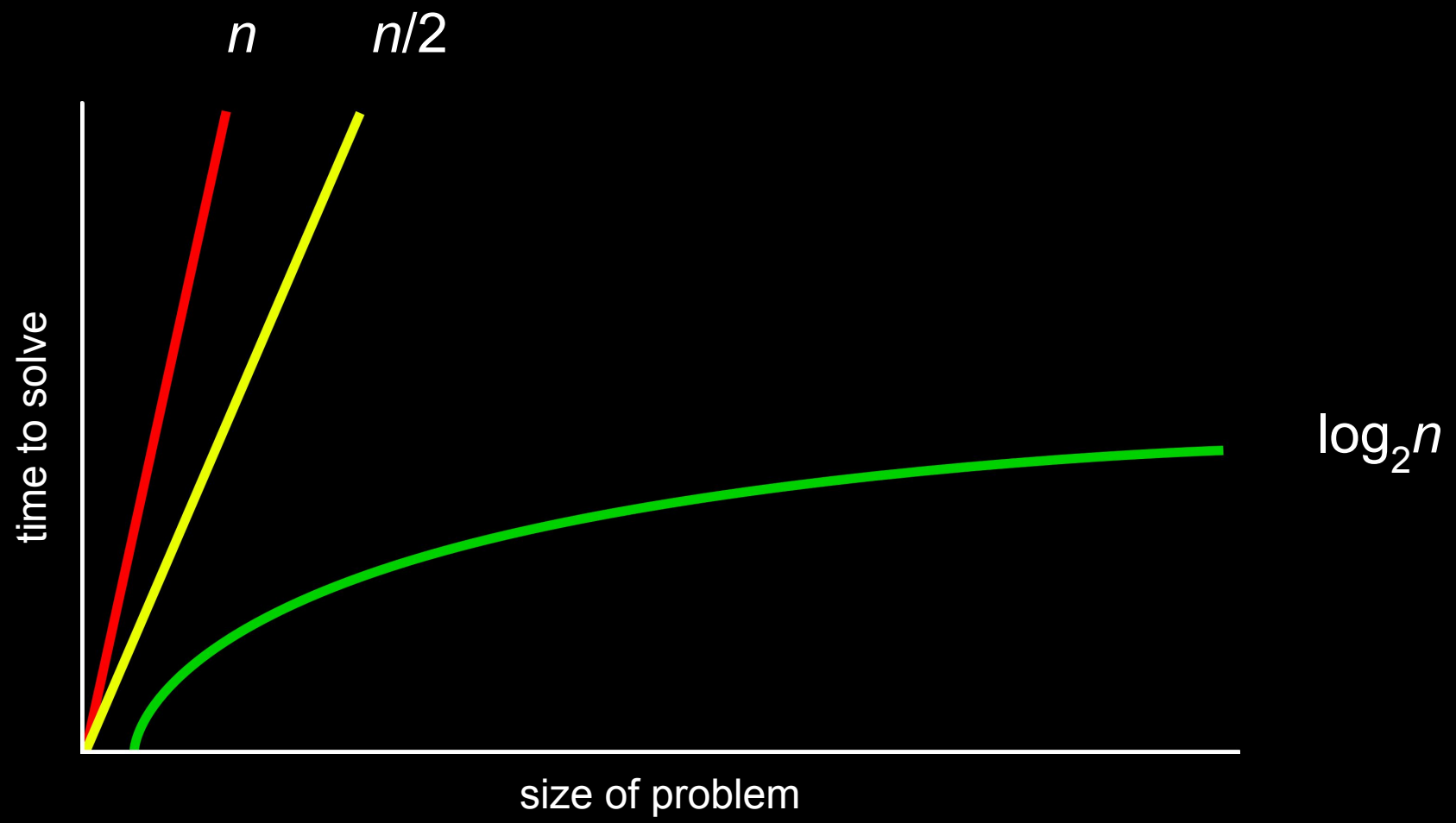

bool

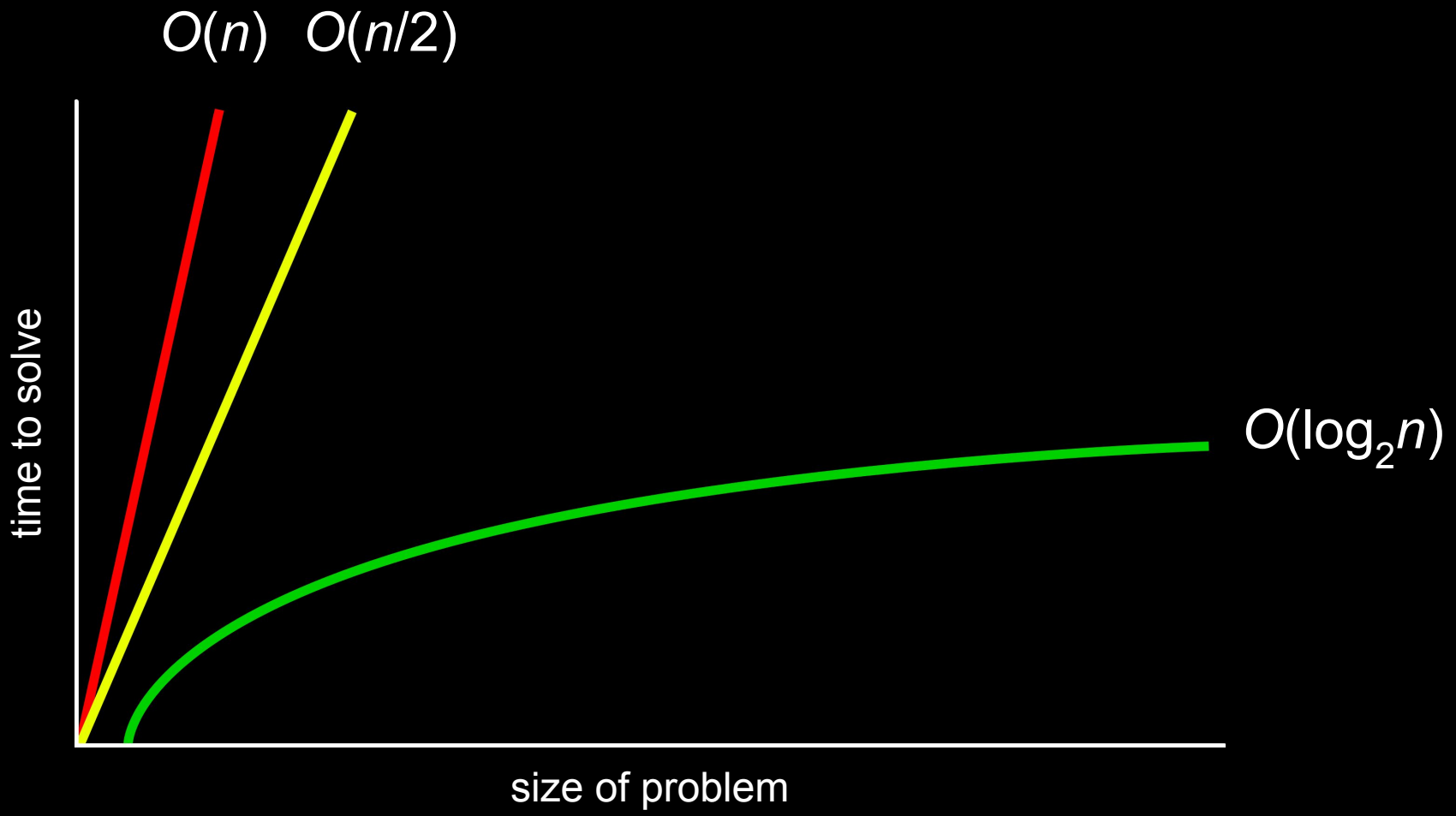

algorithms

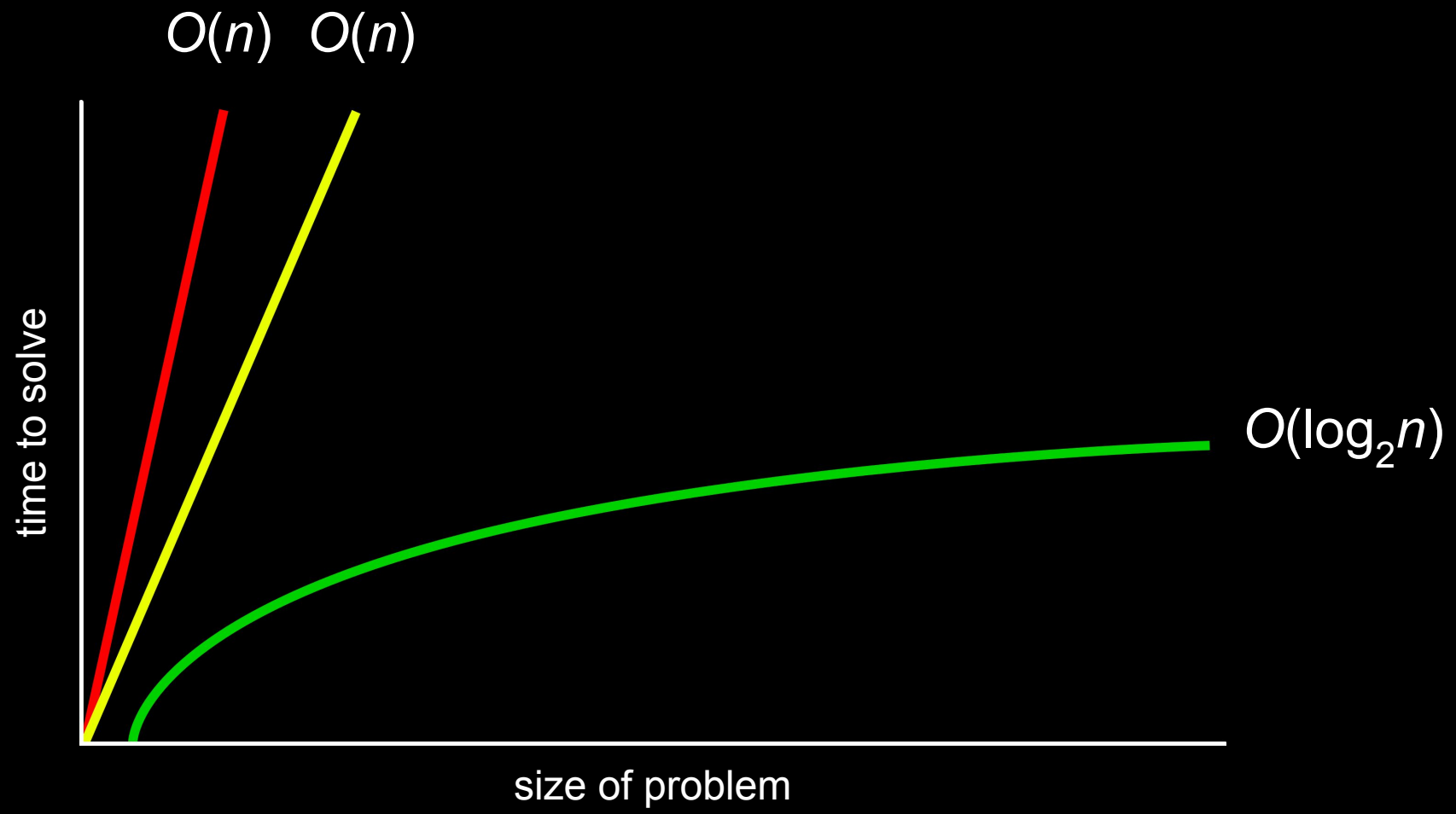
running times

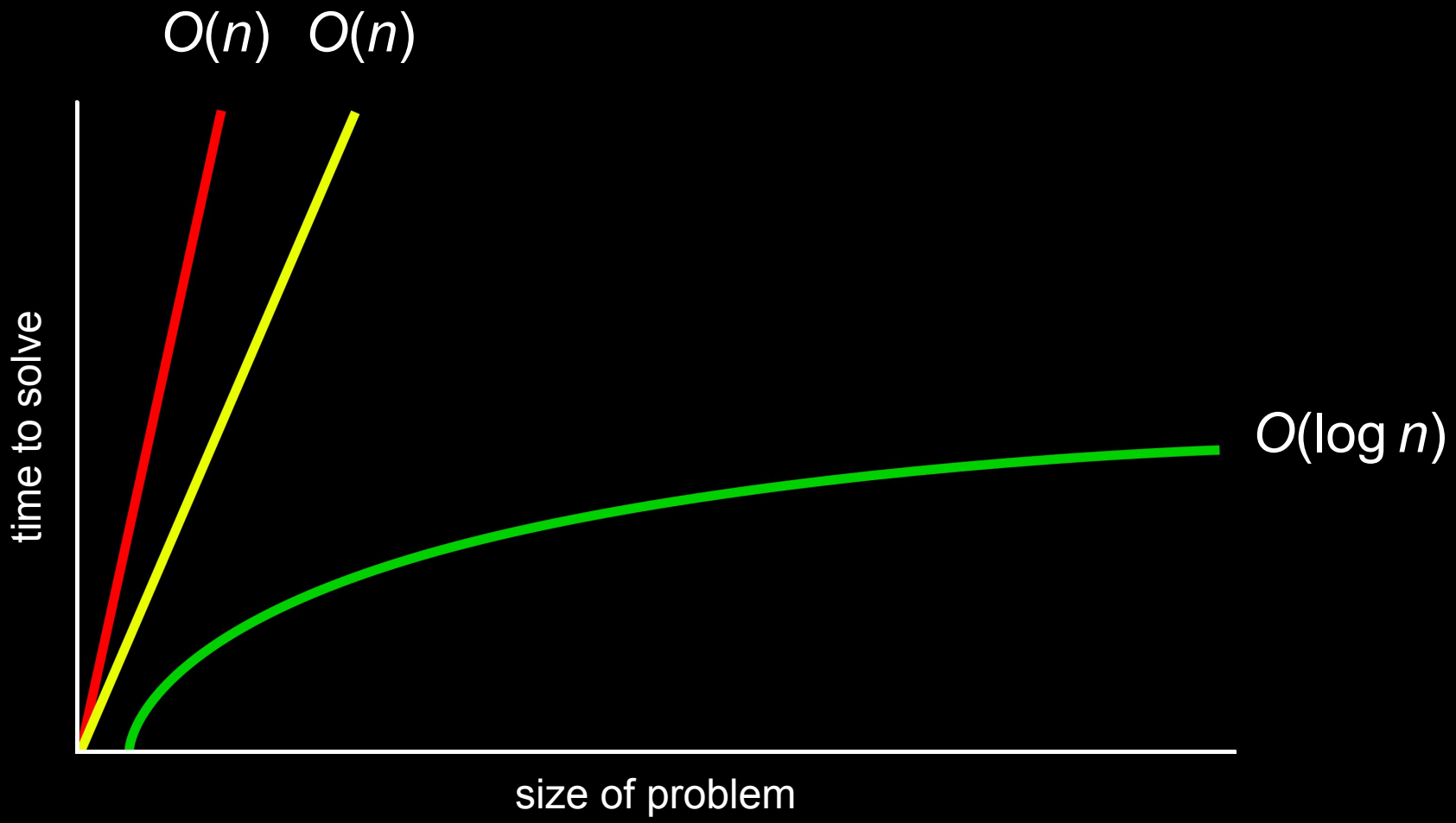
O

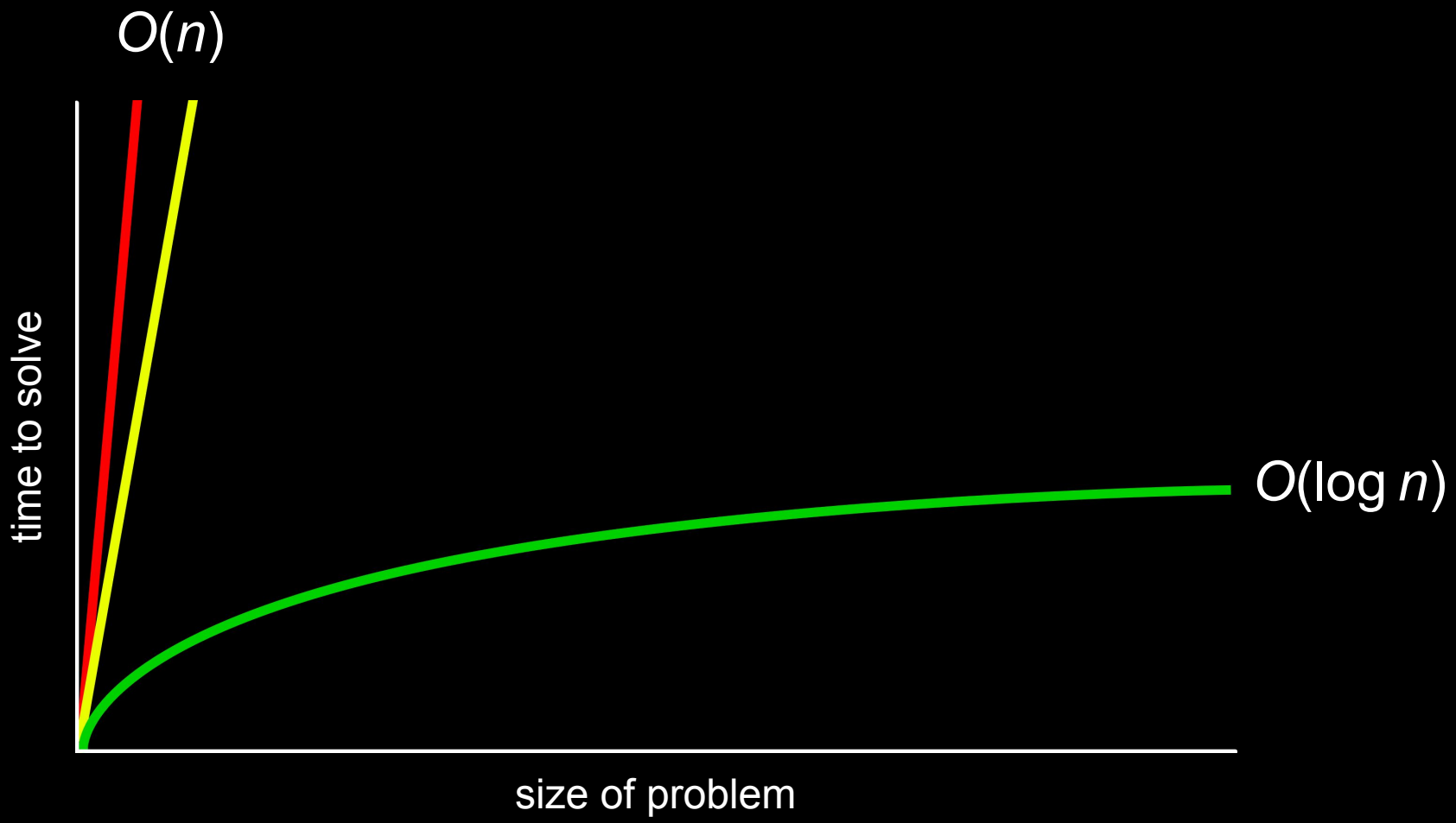

time to solve
size of problem


n
n/2
time to solve
log2n
size of problem


O(n)
O(n/2)
O(log2n)
time to solve
size of problem


O(n)
O(n)
O(log2n)
time to solve
size of problem


O(n)
O(n)
O(log n)
time to solve
size of problem


O(n)
time to solve
O(log n)
size of problem

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

Ω

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$

linear search

```
For i from 0 to n-1
    If number behind i'th door
        Return true
Return false
```

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(n^2)$

$O(n \log n)$

$O(n)$ linear search

$O(\log n)$

$O(1)$

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$ linear search

binary search

```
If number behind middle door
    Return true
Else if number < middle door
    Search left half
Else if number > middle door
    Search right half
```

```
If no doors

If number behind middle door
    Return true
Else if number < middle door
    Search left half
Else if number > middle door
    Search right half
```

```
If no doors
    Return false
If number behind middle door
    Return true
Else if number < middle door
    Search left half
Else if number > middle door
    Search right half
```

$O(n^2)$

$O(n \log n)$

$O(n)$ linear search

$O(\log n)$

$O(1)$

$O(n^2)$

$O(n \log n)$

$O(n)$ linear search

$O(\log n)$ binary search

$O(1)$

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$ linear search

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$ linear search, binary search

```
int numbers[]
```

```
string names[]
```

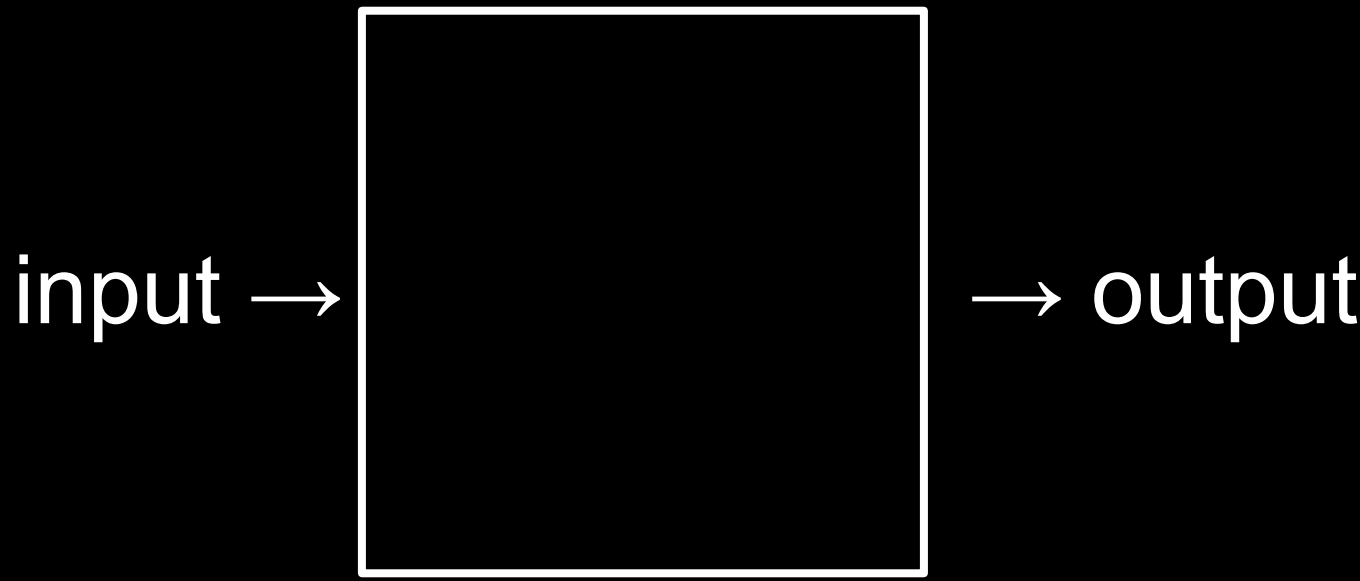
data structures

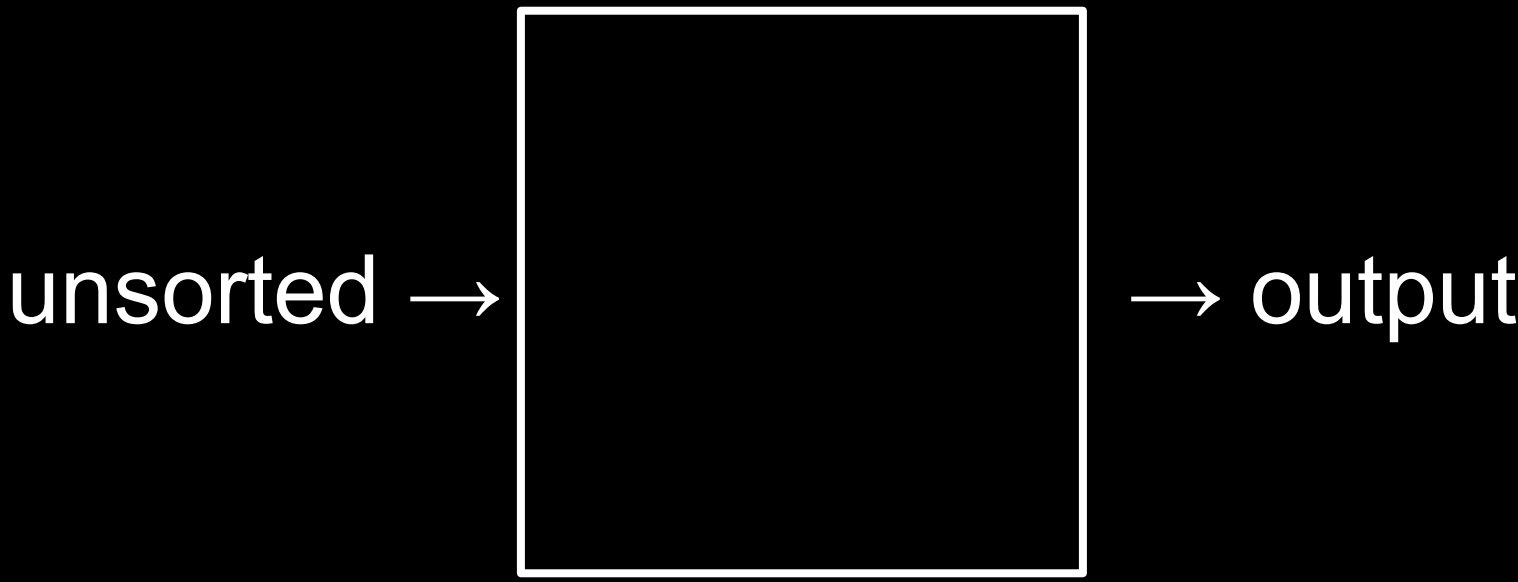
```
person people[]
```

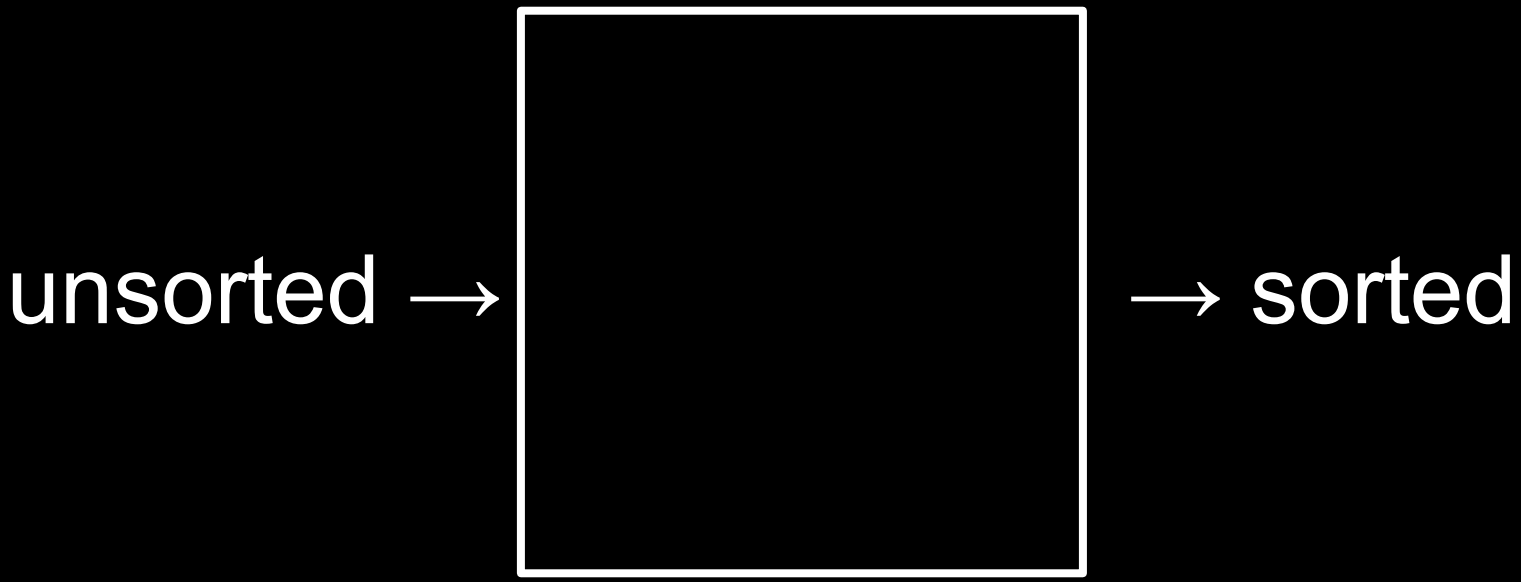
```
string name;
string number;
```

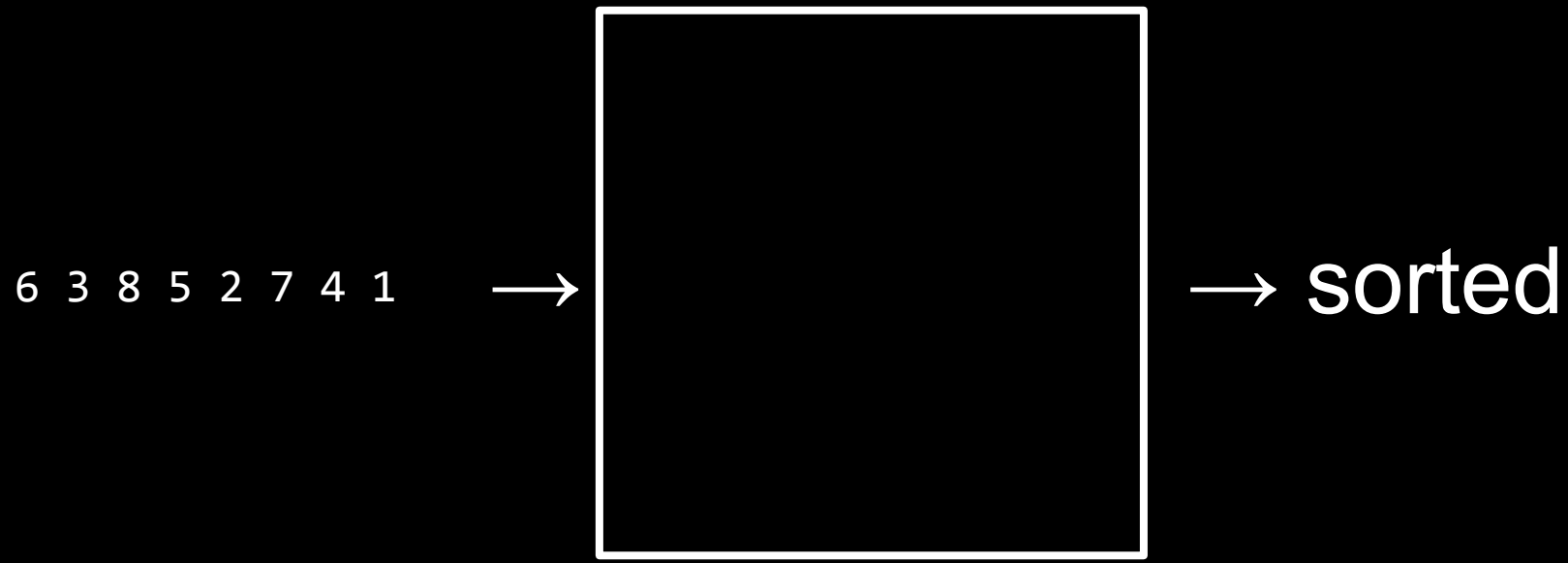
```
typedef struct
{
    string name;
    string number;
}
person;
```

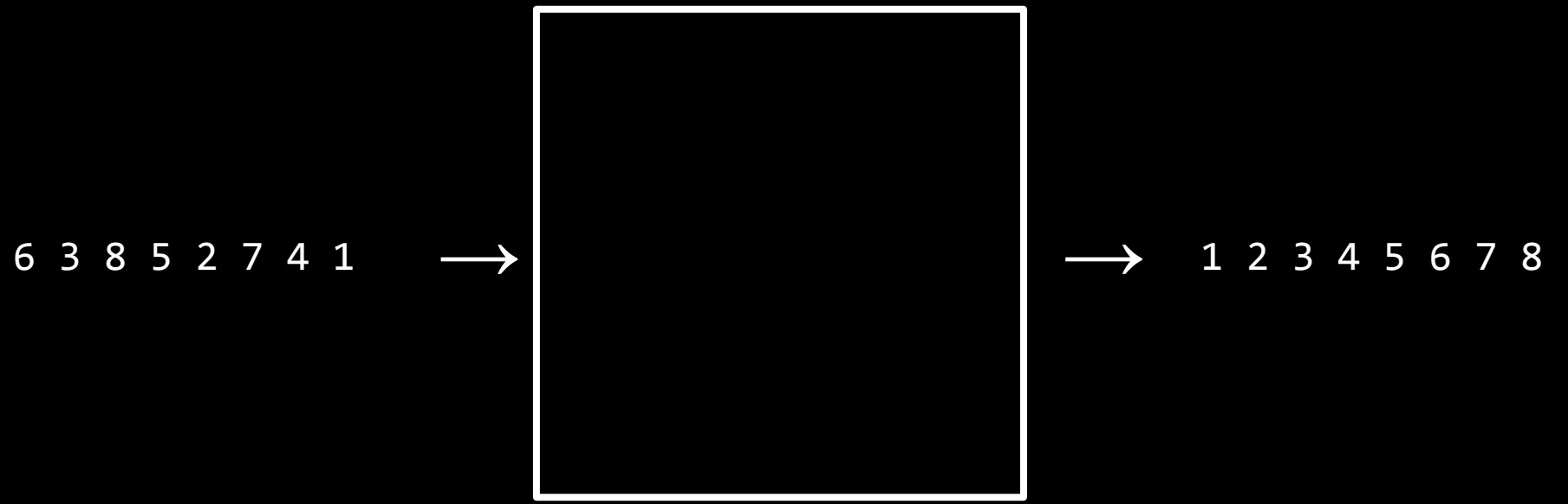
sorting


input →
→ output


unsorted →
→ output


unsorted →
→ sorted


6 3 8 5 2 7 4 1
sorted


6 3 8 5 2 7 4 1
1 2 3 4 5 6 7 8

selection sort

6 3 8 5 2 7 4 1

```
For i from 0 to n-1
    Find smallest item between i'th item and last item
    Swap smallest item with i'th item
```

n

$n + (n - 1)$

$n + (n - 1) + (n - 2)$

$n + (n - 1) + (n - 2) + \ldots + 1$

$n + (n - 1) + (n - 2) + \ldots + 1$

$n(n + 1)/2$

$n + (n - 1) + (n - 2) + \ldots + 1$

$n(n + 1)/2$

$(n^2 + n)/2$

$n + (n - 1) + (n - 2) + \ldots + 1$

$n(n + 1)/2$

$(n^2 + n)/2$

$n^2/2 + n/2$

$n + (n - 1) + (n - 2) + \ldots + 1$

$n(n + 1)/2$

$(n^2 + n)/2$

$n^2/2 + n/2$

$O(n^2)$

$O(n^2)$

$O(n \log n)$

$O(n)$ linear search

$O(\log n)$ binary search

$O(1)$

$O(n^2)$ selection sort

$O(n \log n)$

$O(n)$ linear search

$O(\log n)$ binary search

$O(1)$

```
For i from 0 to n-1
    Find smallest item between i'th item and last item
    Swap smallest item with i'th item
```

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$ linear search, binary search

$\Omega(n^2)$ selection sort

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$ linear search, binary search

bubble sort

6 3 8 5 2 7 4 1

```
Repeat until sorted
    For i from 0 to n-2
        If i'th and i+1'th elements out of order
            Swap them
```

```
Repeat n-1 times
    For i from 0 to n-2
        If i'th and i+1'th elements out of order
            Swap them
```

$(n - 1) \times (n - 1)$

$(n - 1) \times (n - 1)$

$n^2 - 1n - 1n + 1$

$(n - 1) \times (n - 1)$

$n^2 - 1n - 1n + 1$

$n^2 - 2n + 1$

$(n - 1) \times (n - 1)$

$n^2 - 1n - 1n + 1$

$n^2 - 2n + 1$

$O(n^2)$

$O(n^2)$ selection sort

$O(n \log n)$

$O(n)$ linear search

$O(\log n)$ binary search

$O(1)$

$O(n^2)$	selection sort, bubble sort
$O(n \log n)$	
$O(n)$	linear search
$O(\log n)$	binary search
$O(1)$	

```
Repeat n-1 times
    For i from 0 to n-2
        If i'th and i+1'th elements out of order
            Swap them
    If no swaps
        Quit
```

$\Omega(n^2)$ selection sort

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$ linear search, binary search

$\Omega(n^2)$	selection sort
$\Omega(n \log n)$	
$\Omega(n)$	bubble sort
$\Omega(\log n)$	
$\Omega(1)$	linear search, binary search

recursion

```
1   Pick up phone book
2   Open to middle of phone book
3   Look at page
4   If person is on page
5       Call person
6   Else if person is earlier in book
7       Open to middle of left half of book
8       Go back to line 3
9   Else if person is later in book
10      Open to middle of right half of book
11      Go back to line 3
12  Else
13      Quit
```

```
1   Pick up phone book
2   Open to middle of phone book
3   Look at page
4   If person is on page
5       Call person
6   Else if person is earlier in book
7       Open to middle of left half of book
8       Go back to line 3
9   Else if person is later in book
10      Open to middle of right half of book
11      Go back to line 3
12  Else
13      Quit
```

```
1   Pick up phone book
2   Open to middle of phone book
3   Look at page
4   If person is on page
5       Call person
6   Else if person is earlier in book
7       Open to middle of left half of book
8       Go back to line 3
9   Else if person is later in book
10      Open to middle of right half of book
11      Go back to line 3
12  Else
13      Quit
```

```
1   Pick up phone book
2   Open to middle of phone book
3   Look at page
4   If person is on page
5       Call person
6   Else if person is earlier in book
7       Search left half of book
8
9   Else if person is later in book
10      Search right half of book
11
12  Else
13      Quit
```

```
1   Pick up phone book
2   Open to middle of phone book
3   Look at page
4   If person is on page
5       Call person
6   Else if person is earlier in book
7       Search left half of book
8   Else if person is later in book
9       Search right half of book
10  Else
11      Quit
```

merge sort

Sort left half of numbers
Sort right half of numbers
Merge sorted halves

```
If only one number
    Quit
Else
    Sort left half of numbers
    Sort right half of numbers
    Merge sorted halves
```

```
If only one number
    Quit
Else
    Sort left half of numbers
    Sort right half of numbers
    Merge sorted halves
```

3 5 6 8 1 2 4 7

```
If only one number
    Quit
Else
    Sort left half of numbers
    Sort right half of numbers
    Merge sorted halves
```

$O(n^2)$	selection sort, bubble sort
$O(n \log n)$	
$O(n)$	linear search
$O(\log n)$	binary search
$O(1)$	

$O(n^2)$	selection sort, bubble sort
$O(n \log n)$	merge sort
$O(n)$	linear search
$O(\log n)$	binary search
$O(1)$	

$\Omega(n^2)$	selection sort
$\Omega(n \log n)$	
$\Omega(n)$	bubble sort
$\Omega(\log n)$	
$\Omega(1)$	linear search, binary search

$\Omega(n^2)$	selection sort
$\Omega(n \log n)$	merge sort
$\Omega(n)$	bubble sort
$\Omega(\log n)$	
$\Omega(1)$	linear search, binary search

θ

$\Theta(n^2)$

$\Theta(n \log n)$

$\Theta(n)$

$\Theta(\log n)$

$\Theta(1)$

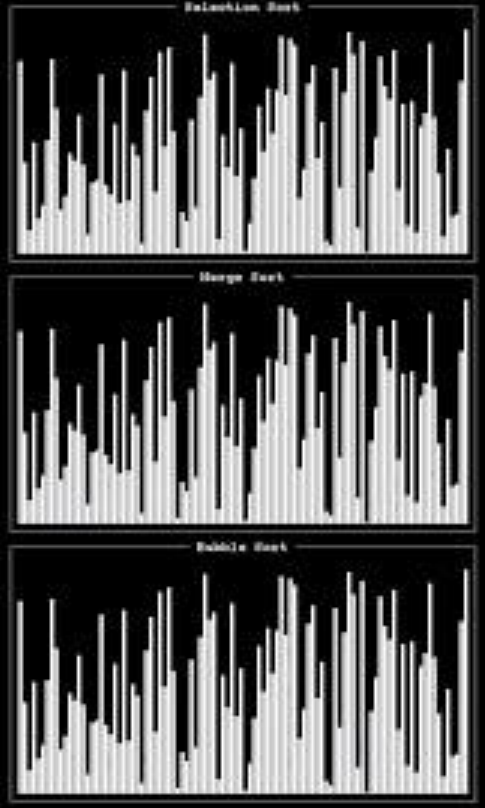
$\Theta(n^2)$ selection sort

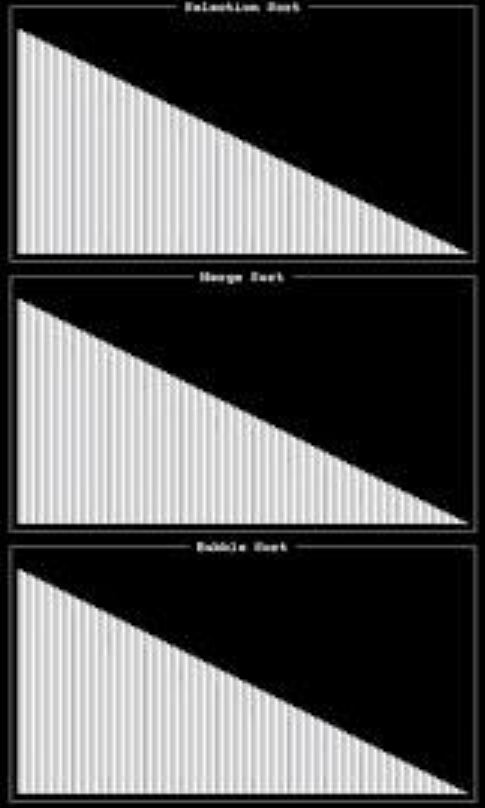
$\Theta(n \log n)$ merge sort

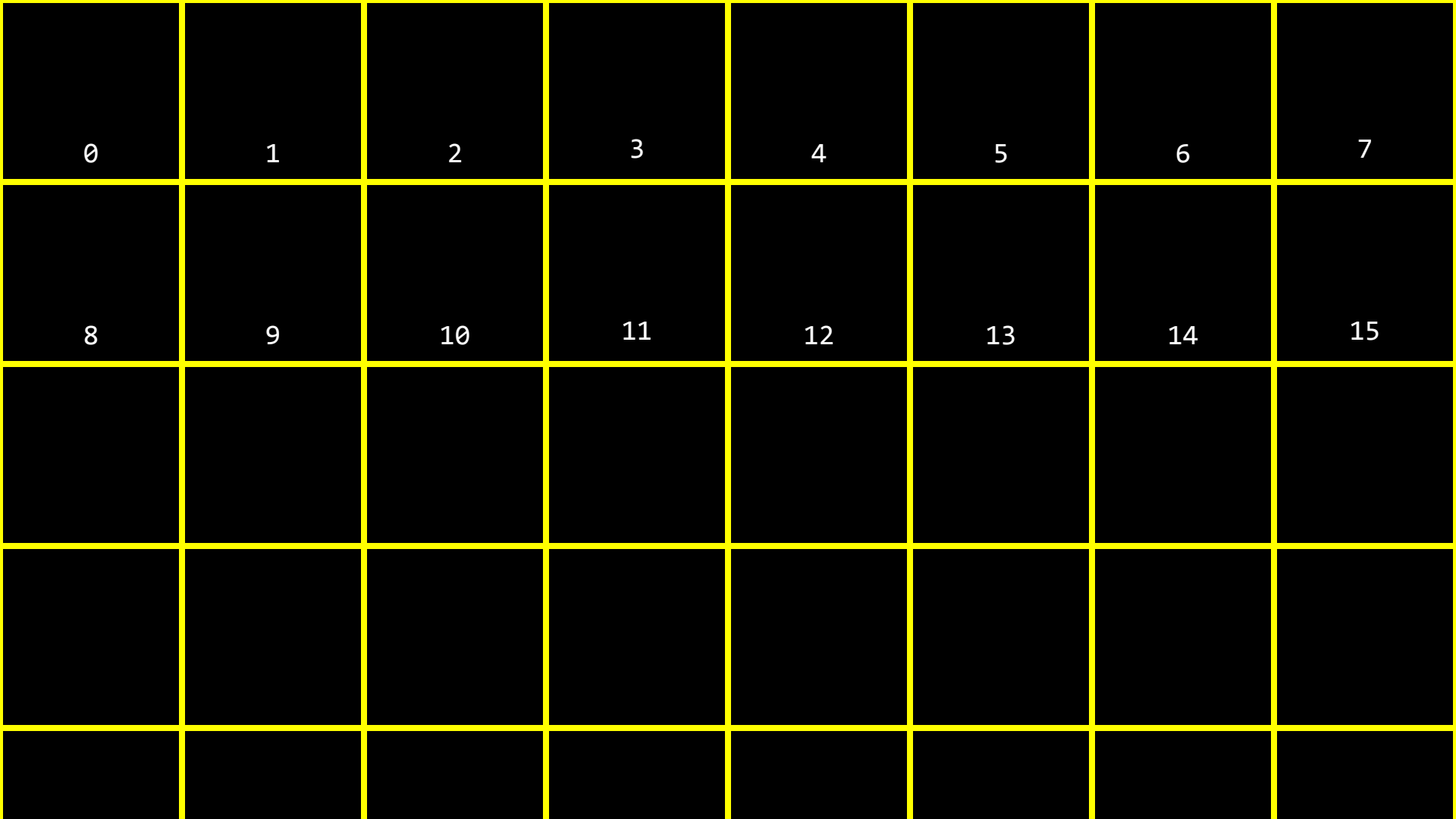
$\Theta(n)$

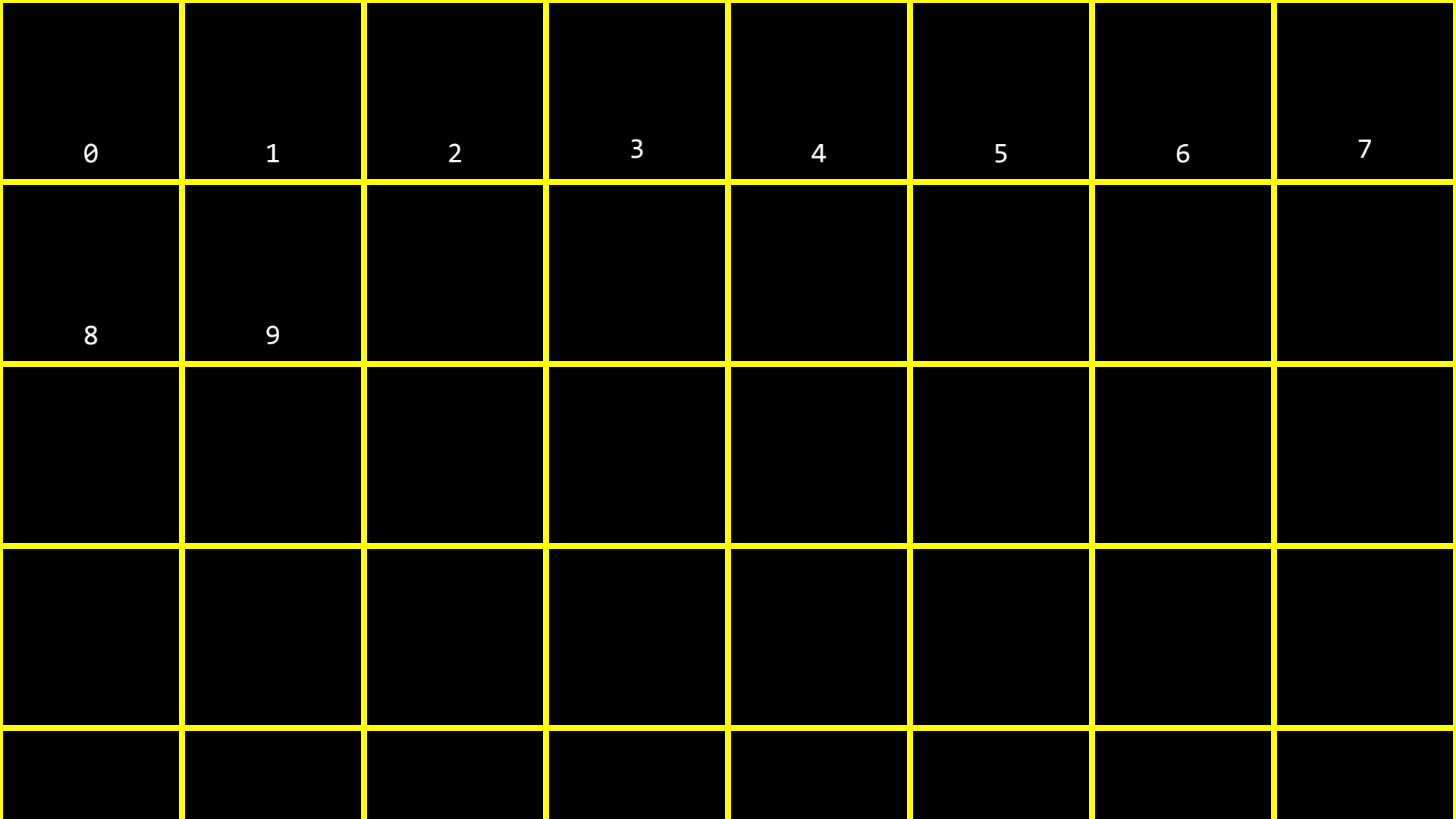
$\Theta(\log n)$

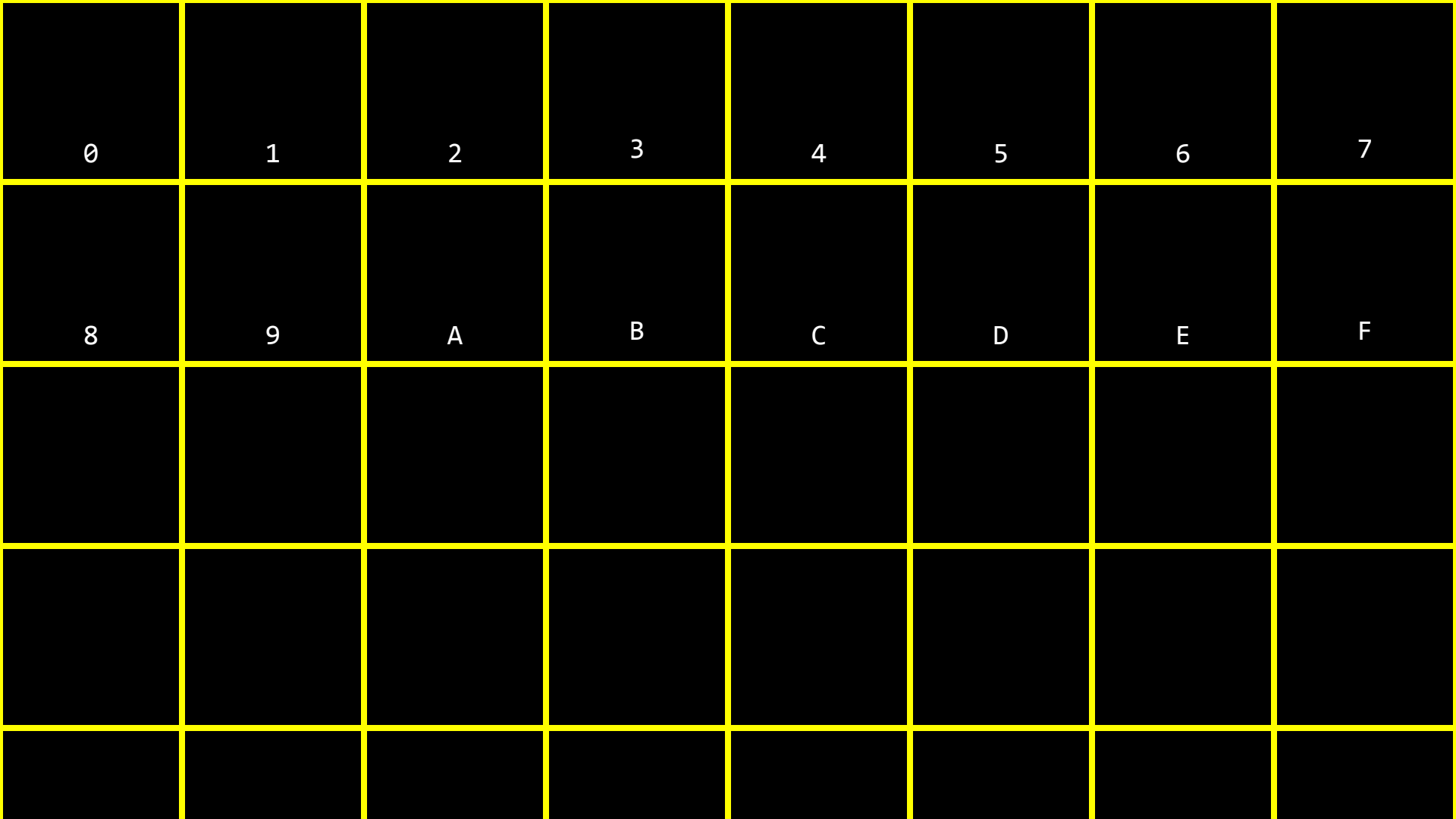
$\Theta(1)$


Unknown Band
Unknown Band
Unknown Band


Selection Sort
Merge Sort
Bubble Sort


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


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

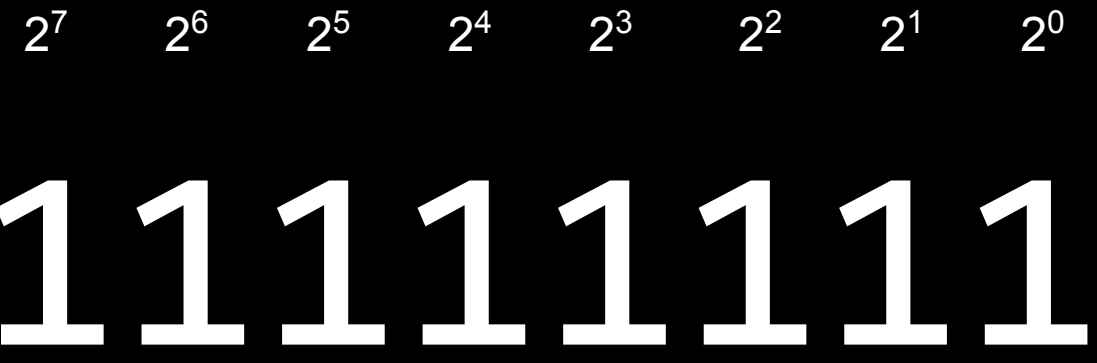
0 1

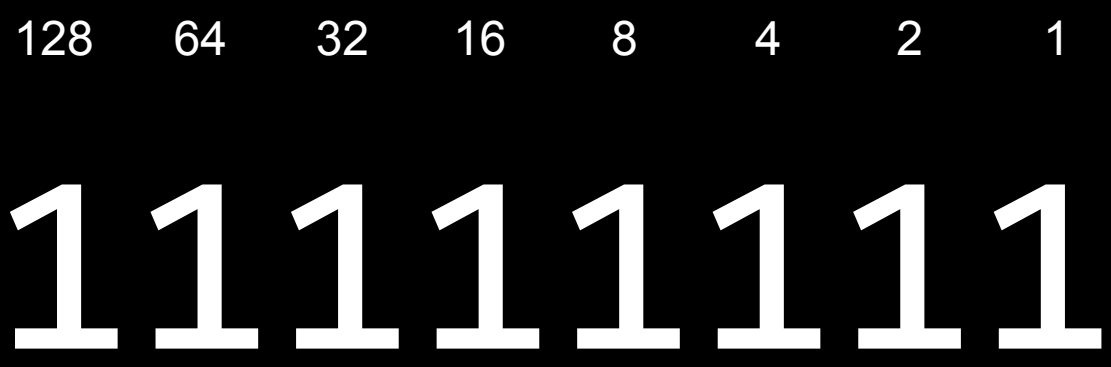
0 1 2 3 4 5 6 7 8 9

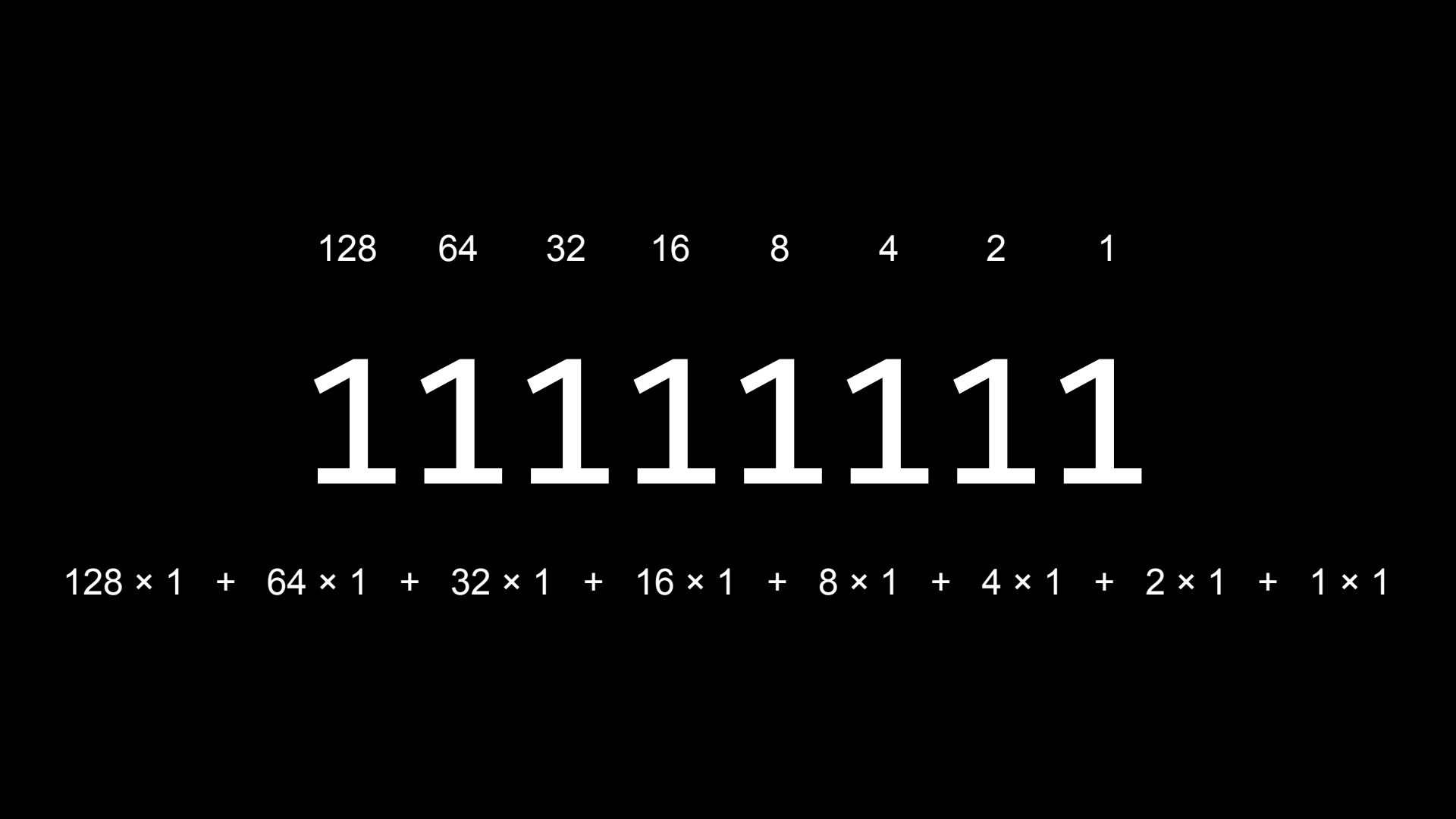
0 1 2 3 4 5 6 7 8 9 A B C D E F

base-16

hexadecimal


2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
11111111


128 64 32 16 8 4 2 1
11111111


128 64 32 16 8 4 2 1
11111111
128 × 1 + 64 × 1 + 32 × 1 + 16 × 1 + 8 × 1 + 4 × 1 + 2 × 1 + 1 × 1

128 64 32 16 8 4 2 1

11111111

255

10^2 10^1 10^0

255

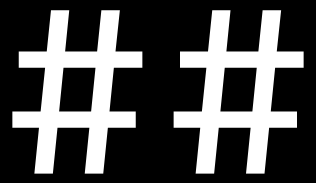
100 10 1

255

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

0B

16 1

0C

16 1

0D

16 1

0E

16 1

0F

16 1

10

16 1

16 1

FF

16 1

F F

16 × F + 1 × F

16 1

F F

16 × 15 + 1 × 15

16 1

F F

240 + 15

16 1

255

128 64 32 16 8 4 2 1

11111111

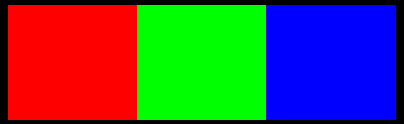
255

11111111

1111 1111

F F

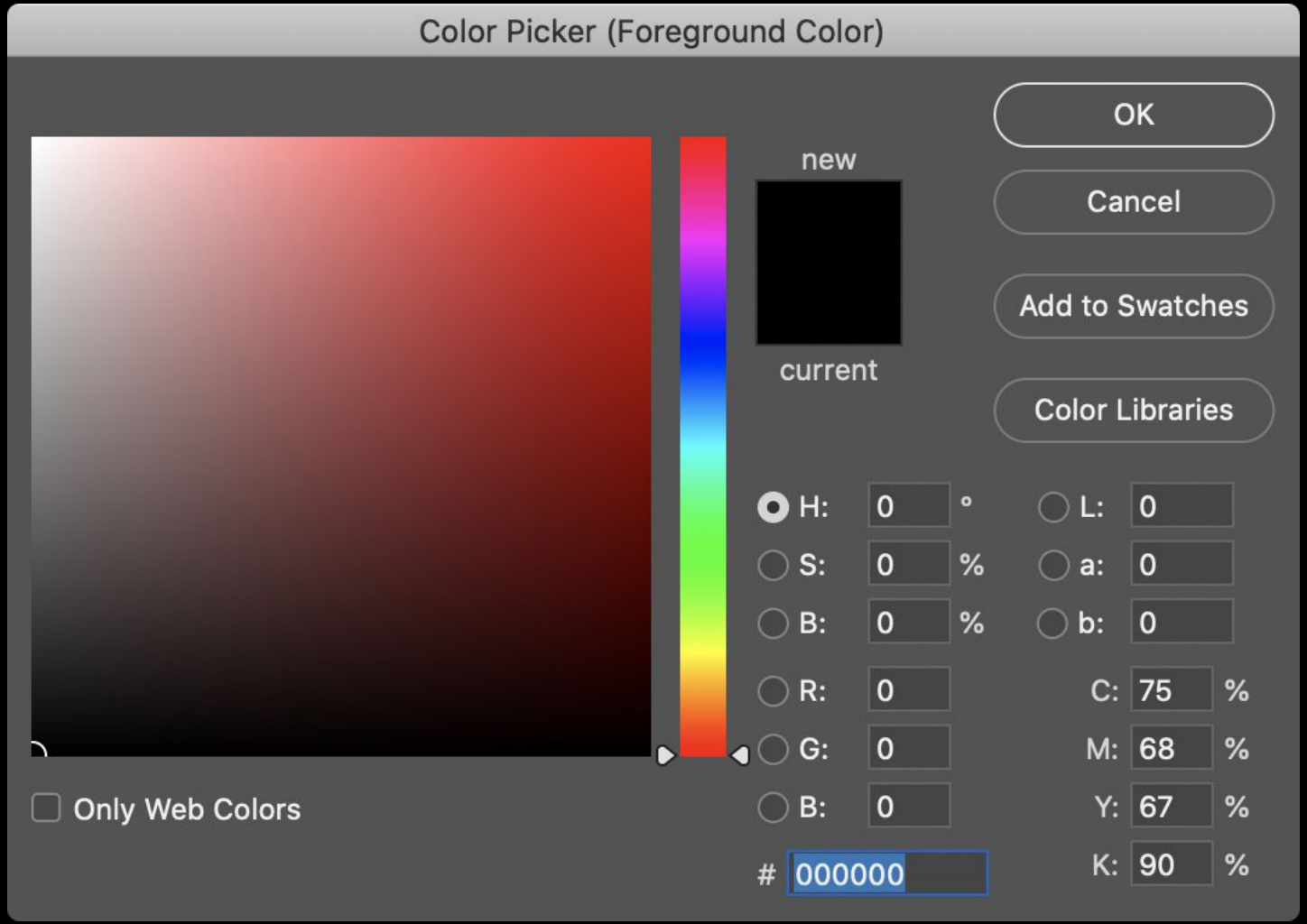
RGB

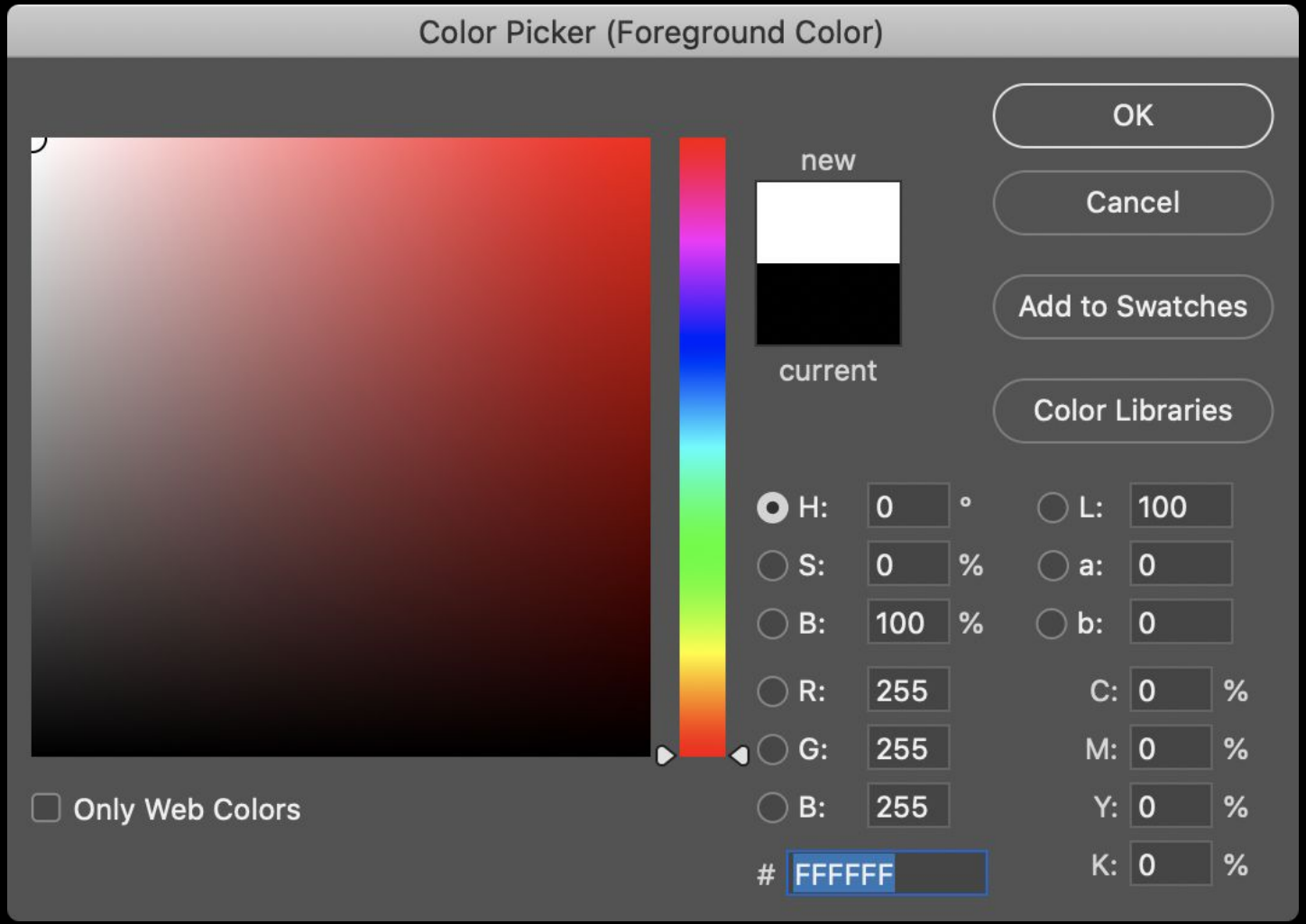


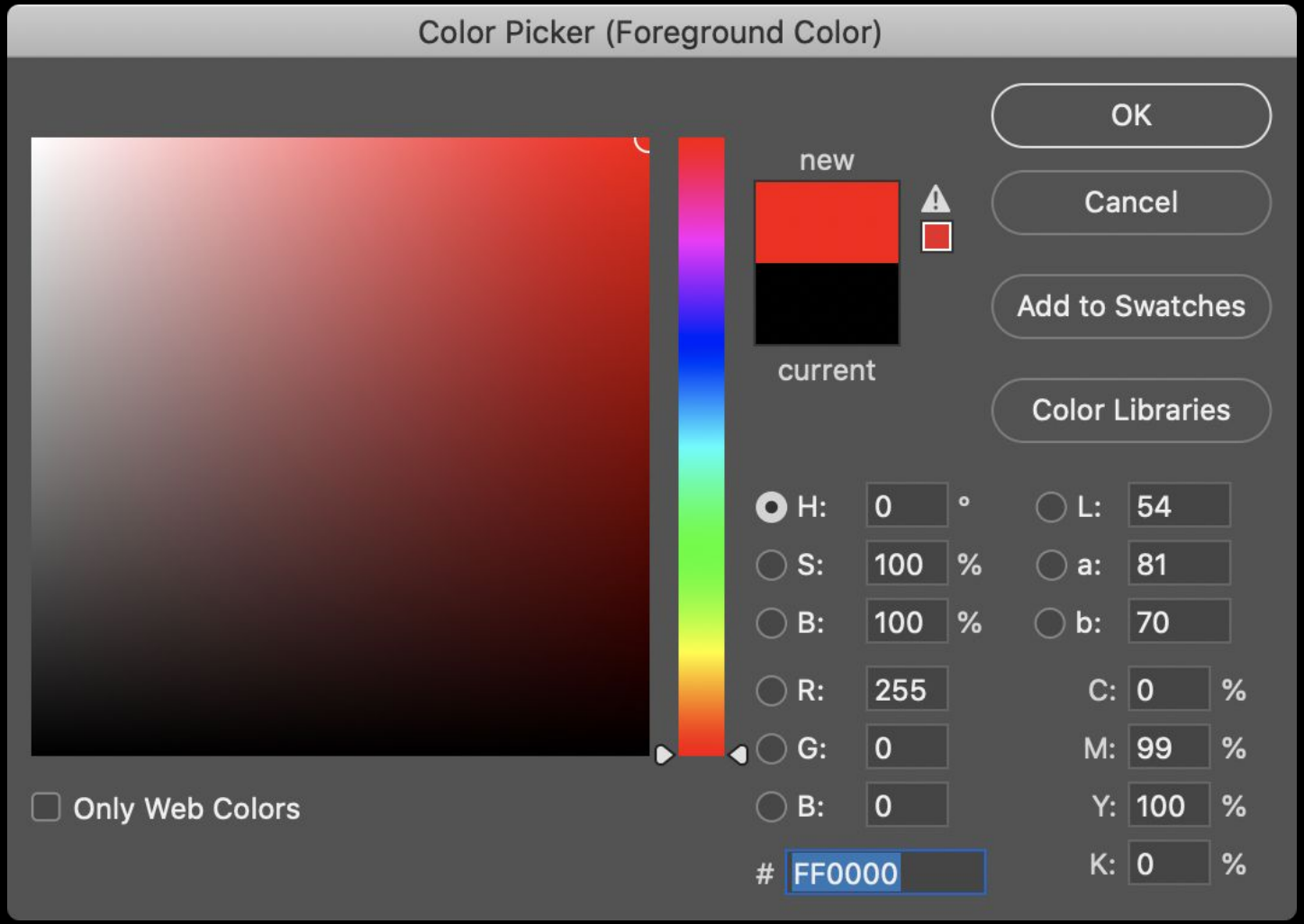

72
73
33

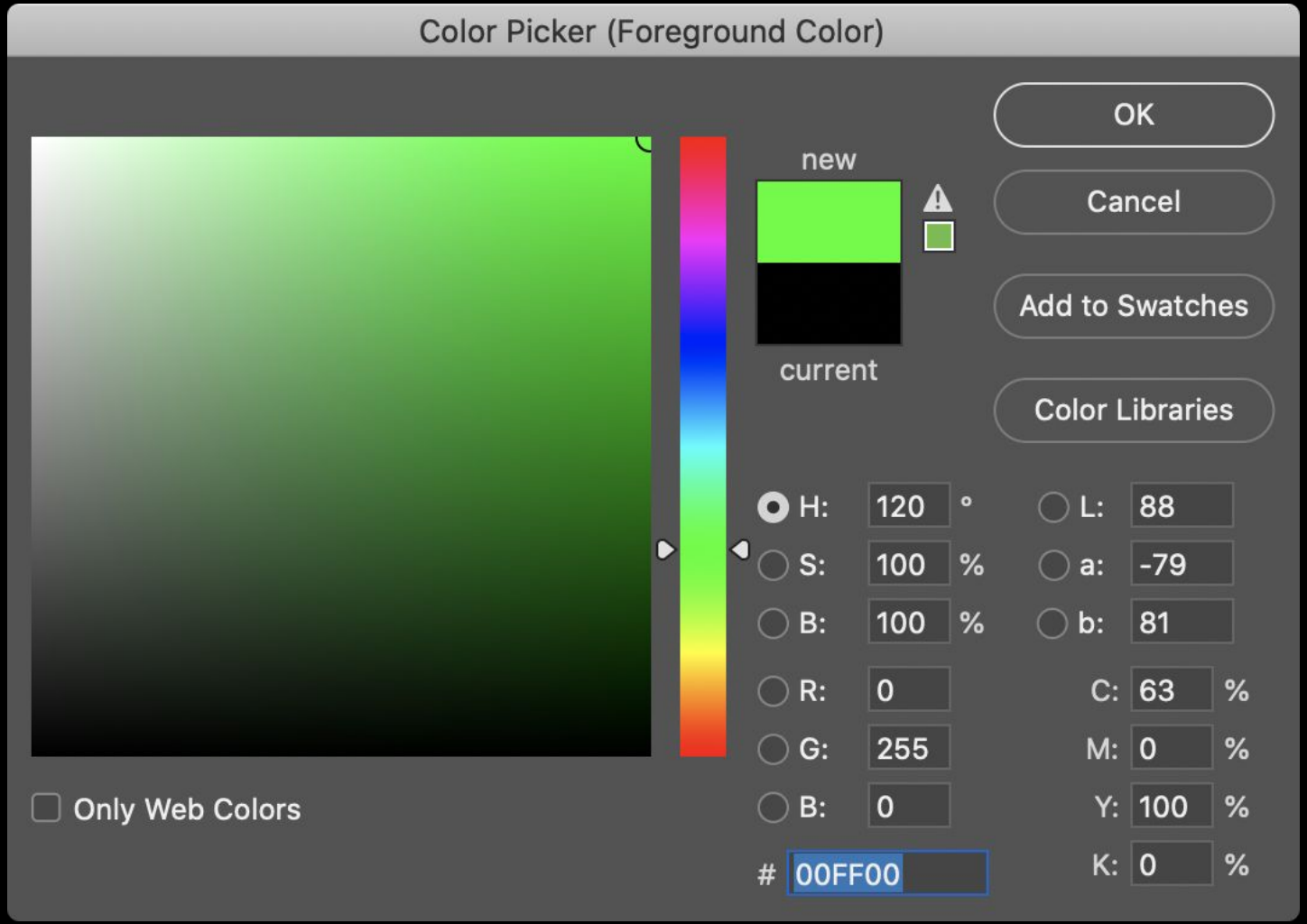

48
49
21

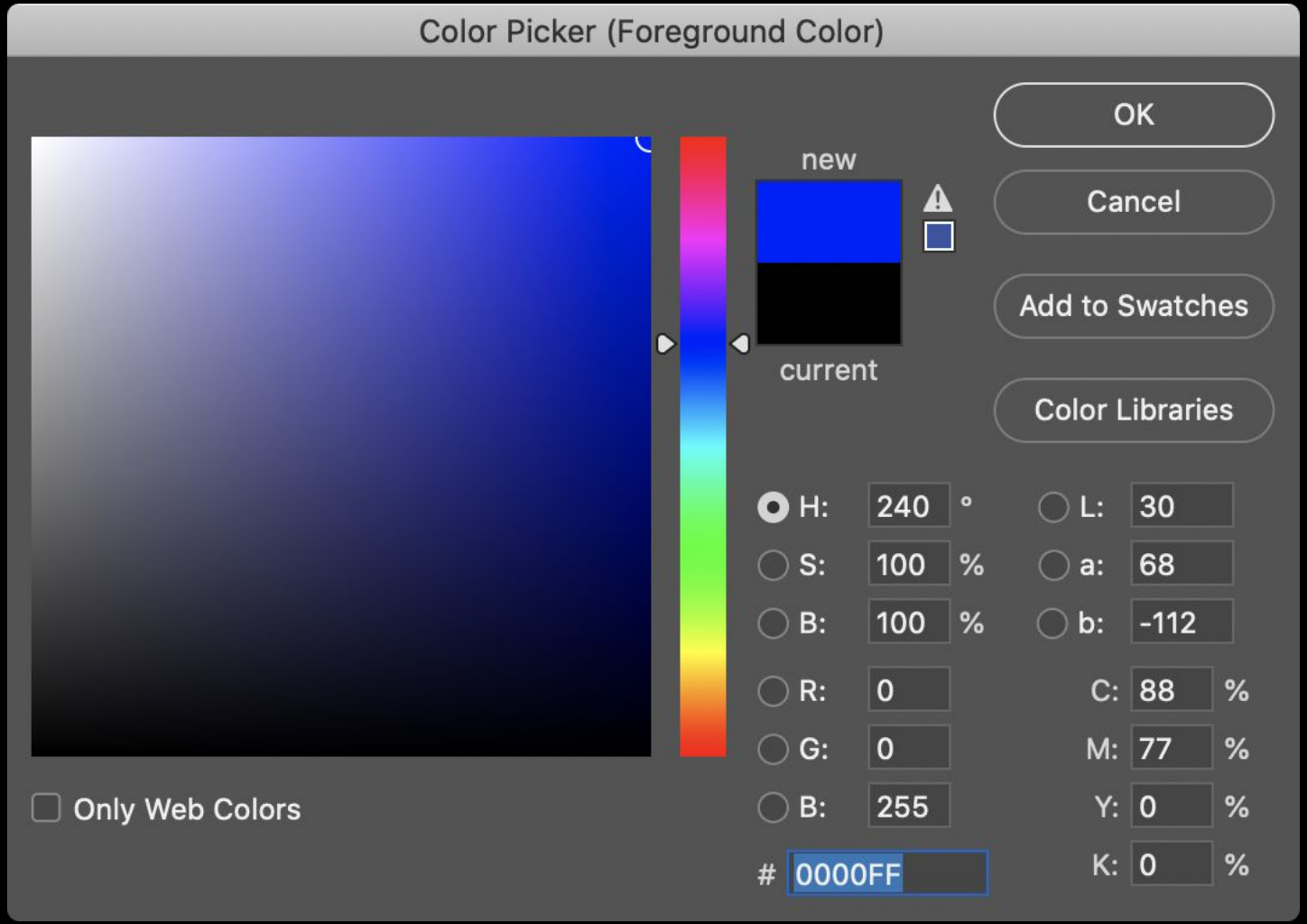

0x48
0x49
0x21

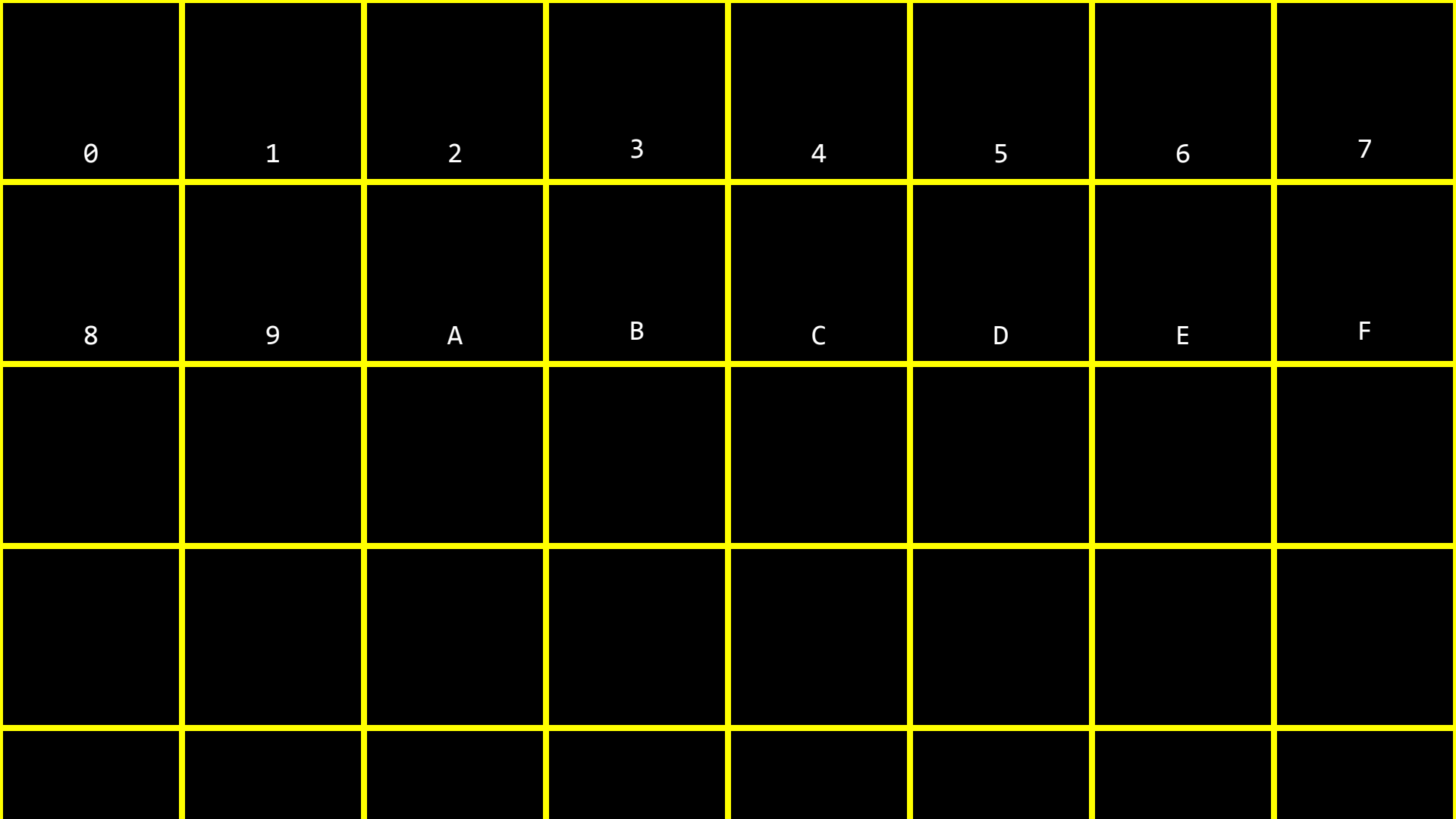

Color Picker (Foreground Color)
OK
Cancel
Add to Swatches
Color Libraries
new
current
H: 0 °
S: 0 %
B: 0 %
R: 0
G: 0
B: 0
L: 0
a: 0
b: 0
C: 75 %
M: 68 %
Y: 67 %
K: 90 %
000000
Only Web Colors


Color Picker (Foreground Color)
OK
Cancel
Add to Swatches
Color Libraries
new
current
H: 0 °
S: 0 %
B: 100 %
R: 255
G: 255
B: 255
FFFFFF
L: 100
a: 0
b: 0
C: 0 %
M: 0 %
Y: 0 %
K: 0 %
Only Web Colors

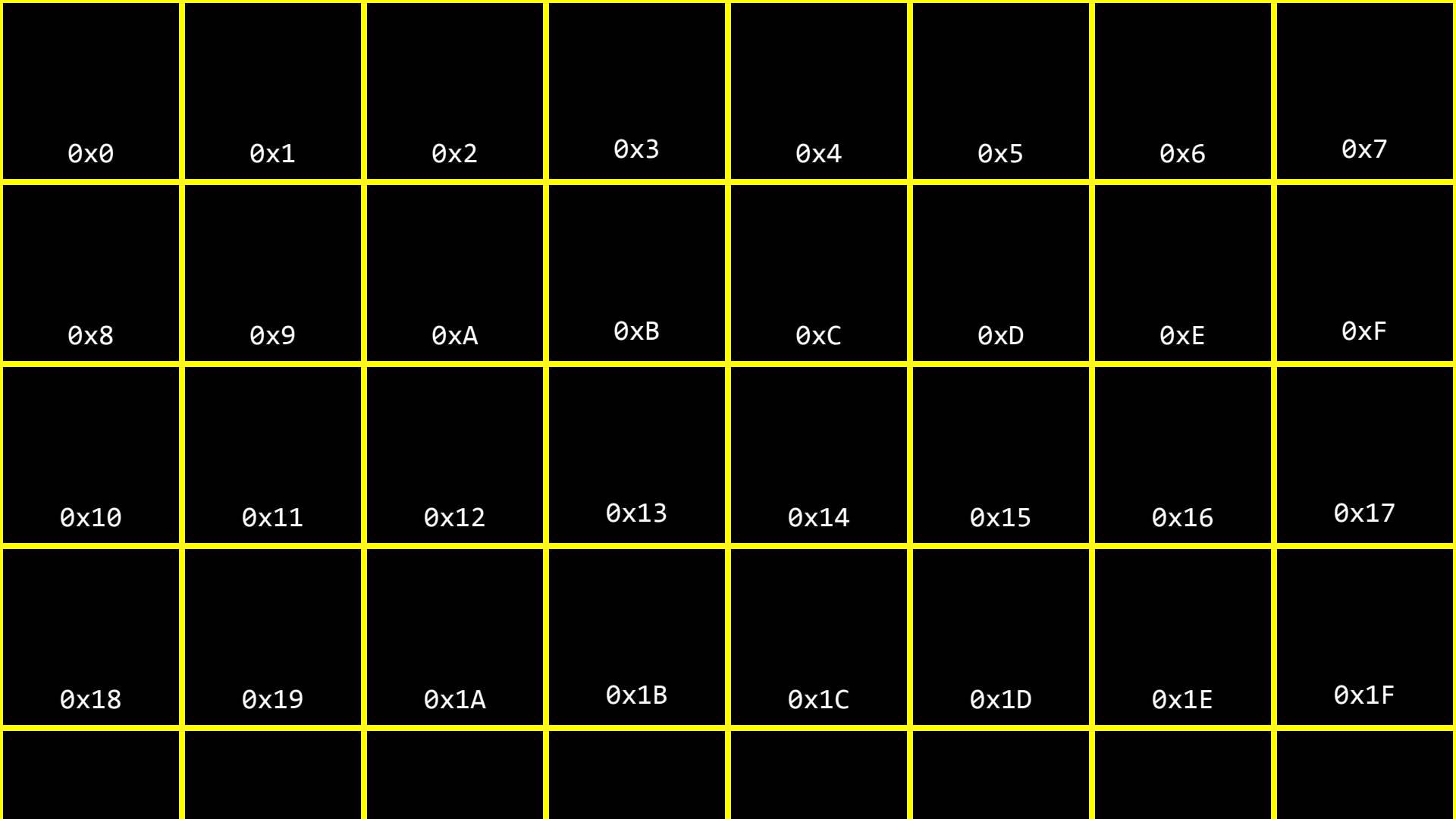

Color Picker (Foreground Color)
OK
Cancel
Add to Swatches
Color Libraries
new
current
H: 0 °
S: 100 %
B: 100 %
R: 255
G: 0
B: 0
FF0000
L: 54
a: 81
b: 70
C: 0 %
M: 99 %
Y: 100 %
K: 0 %
Only Web Colors


Color Picker (Foreground Color)
OK
Cancel
Add to Swatches
Color Libraries
new
current
H: 120 °
S: 100 %
B: 100 %
R: 0
G: 255
B: 0
00FF00
L: 88
a: -79
b: 81
C: 63 %
M: 0 %
Y: 100 %
K: 0 %
Only Web Colors

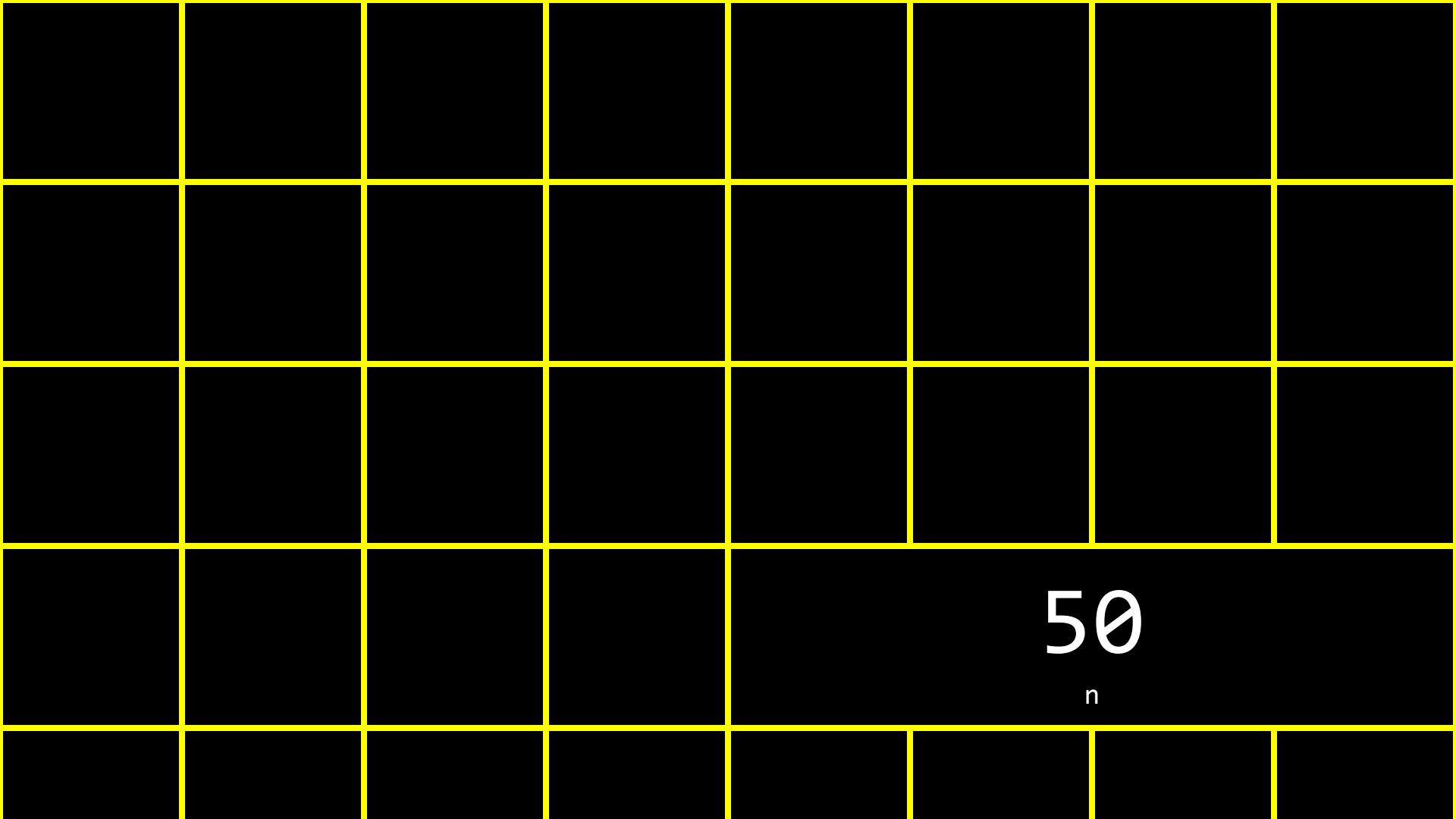

Color Picker (Foreground Color)
OK
Cancel
Add to Swatches
Color Libraries
new
current
H: 240 °
S: 100 %
B: 100 %
R: 0
G: 0
B: 255
0000FF
L: 30
a: 68
b: -112
C: 88 %
M: 77 %
Y: 0 %
K: 0 %
Only Web Colors

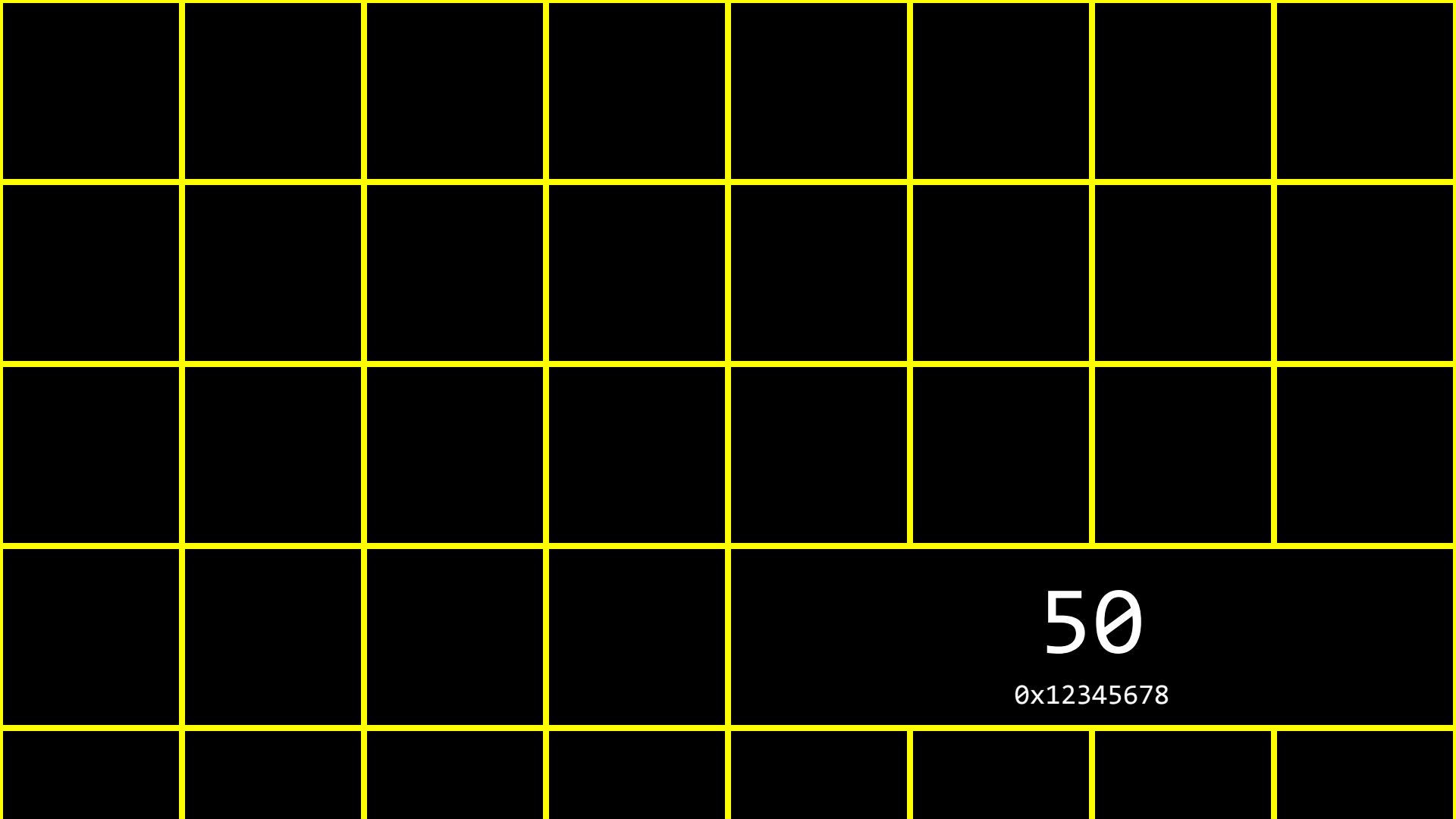

0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F

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

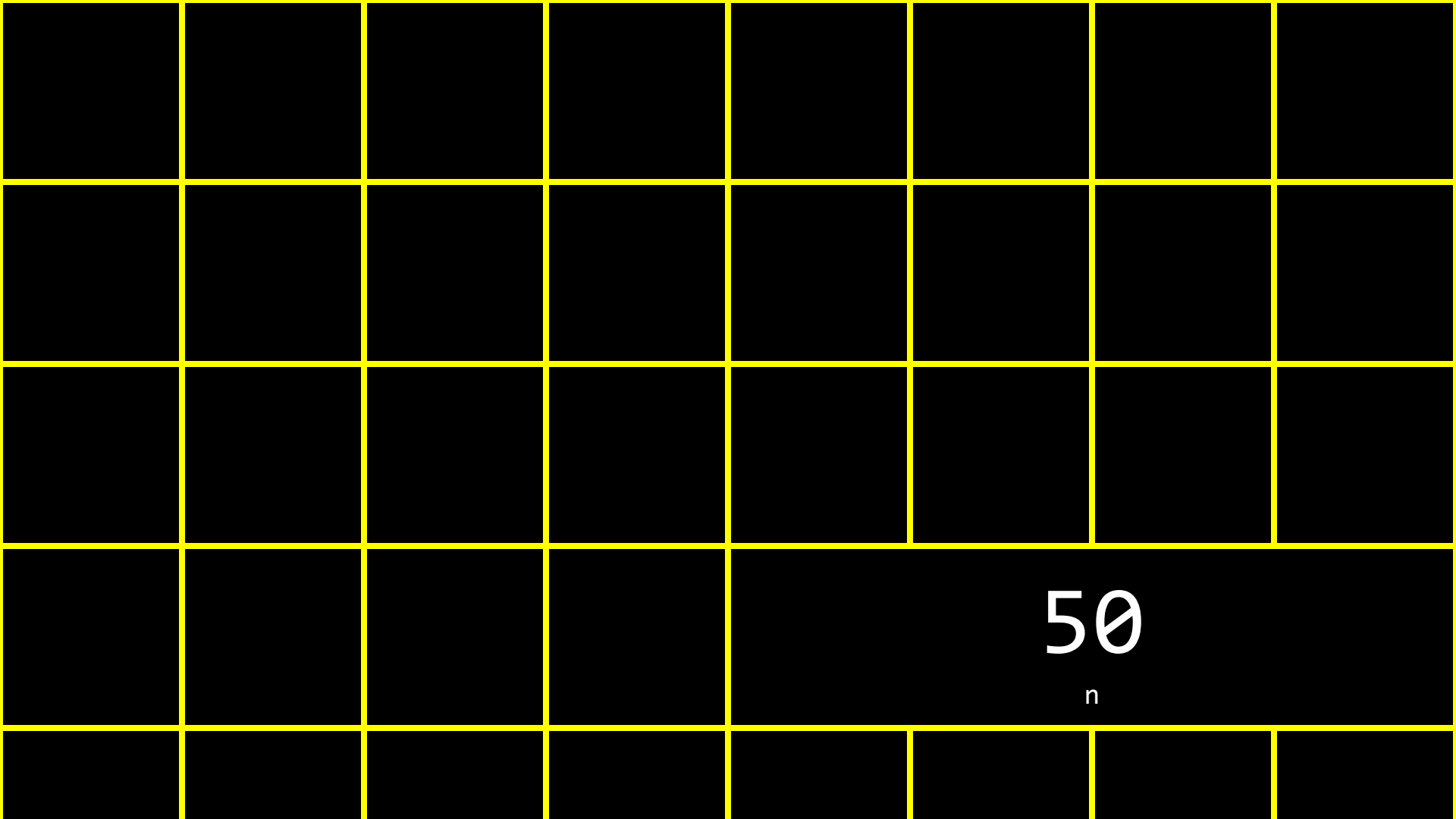

50
0x12345678

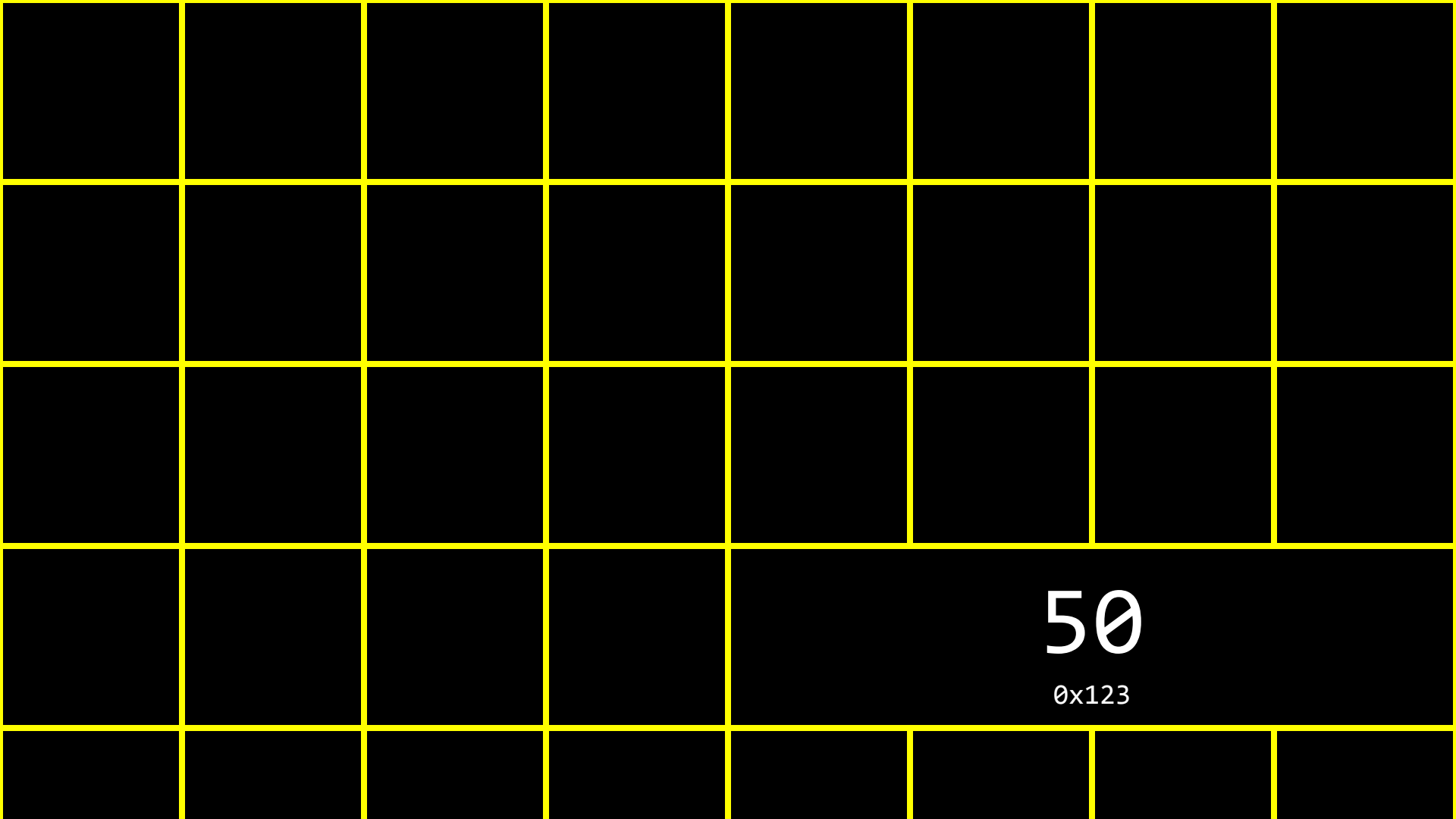
&

*

pointers

```
int n = 50;

int *p = &n;
```


50
n


50
0x123

0x123

p

50

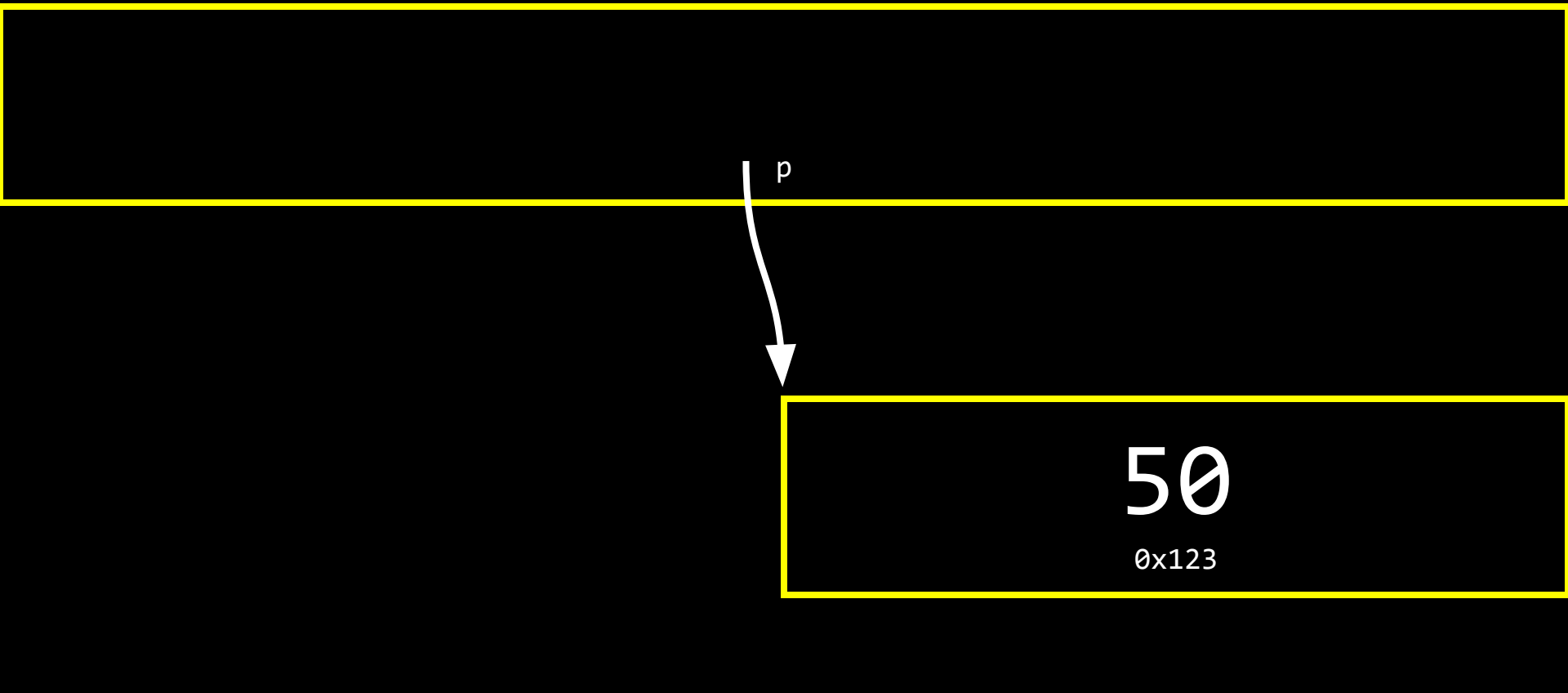
0x123

0x123

p

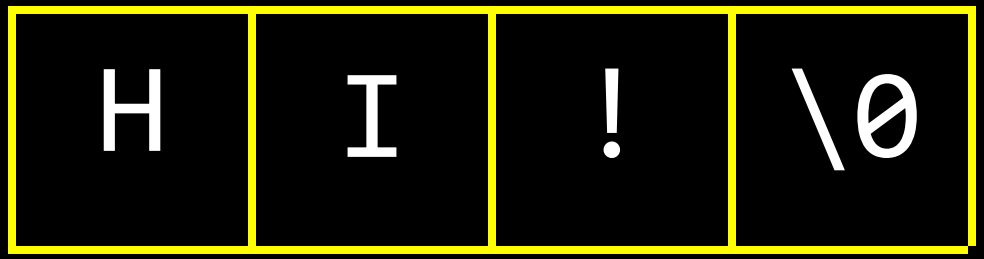
50

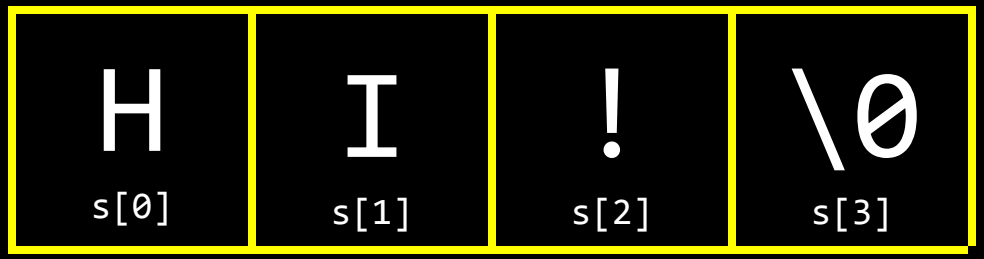
0x123

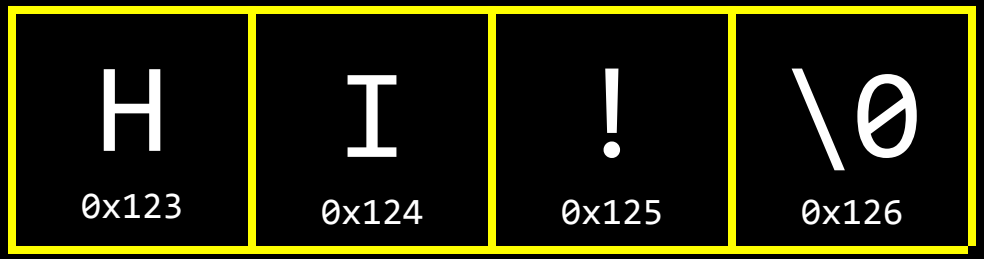

p
50
0x123

string

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

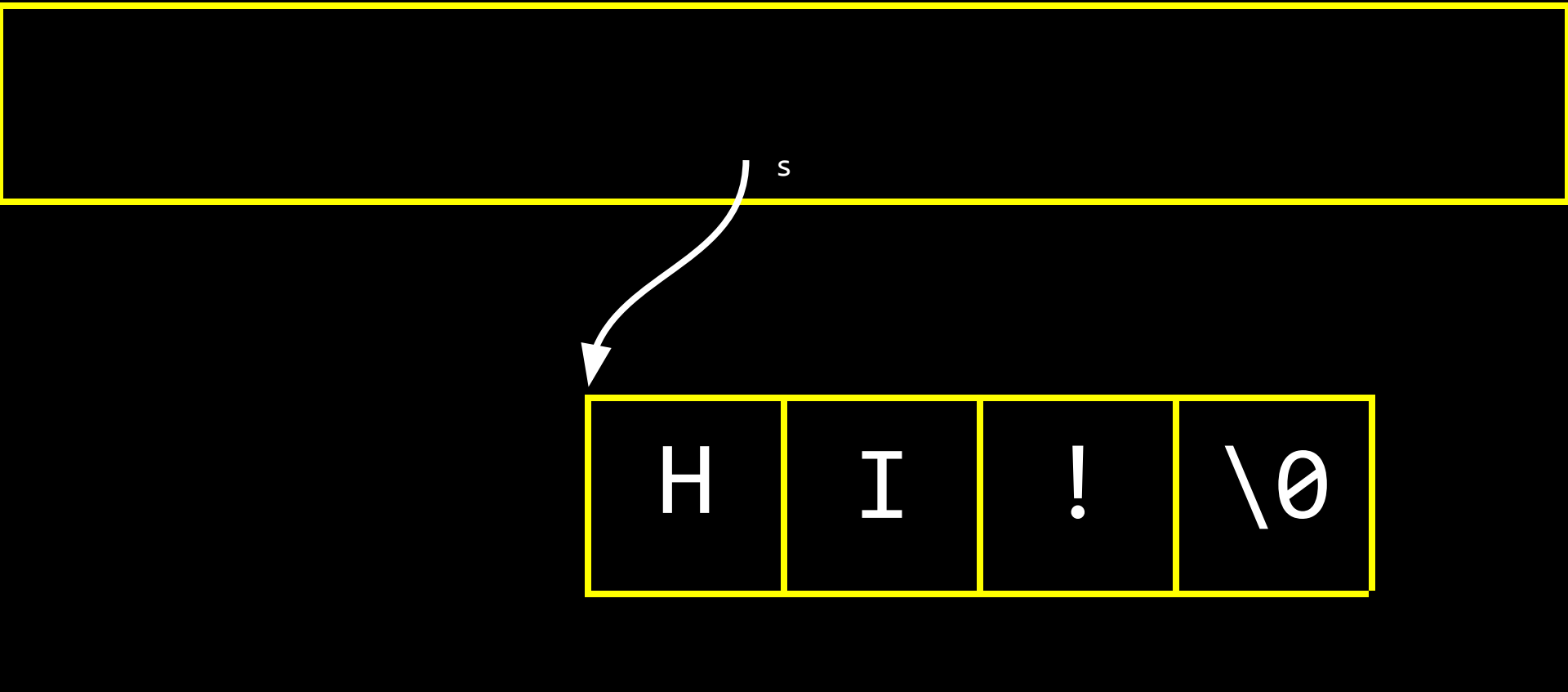

H
I
!
\0

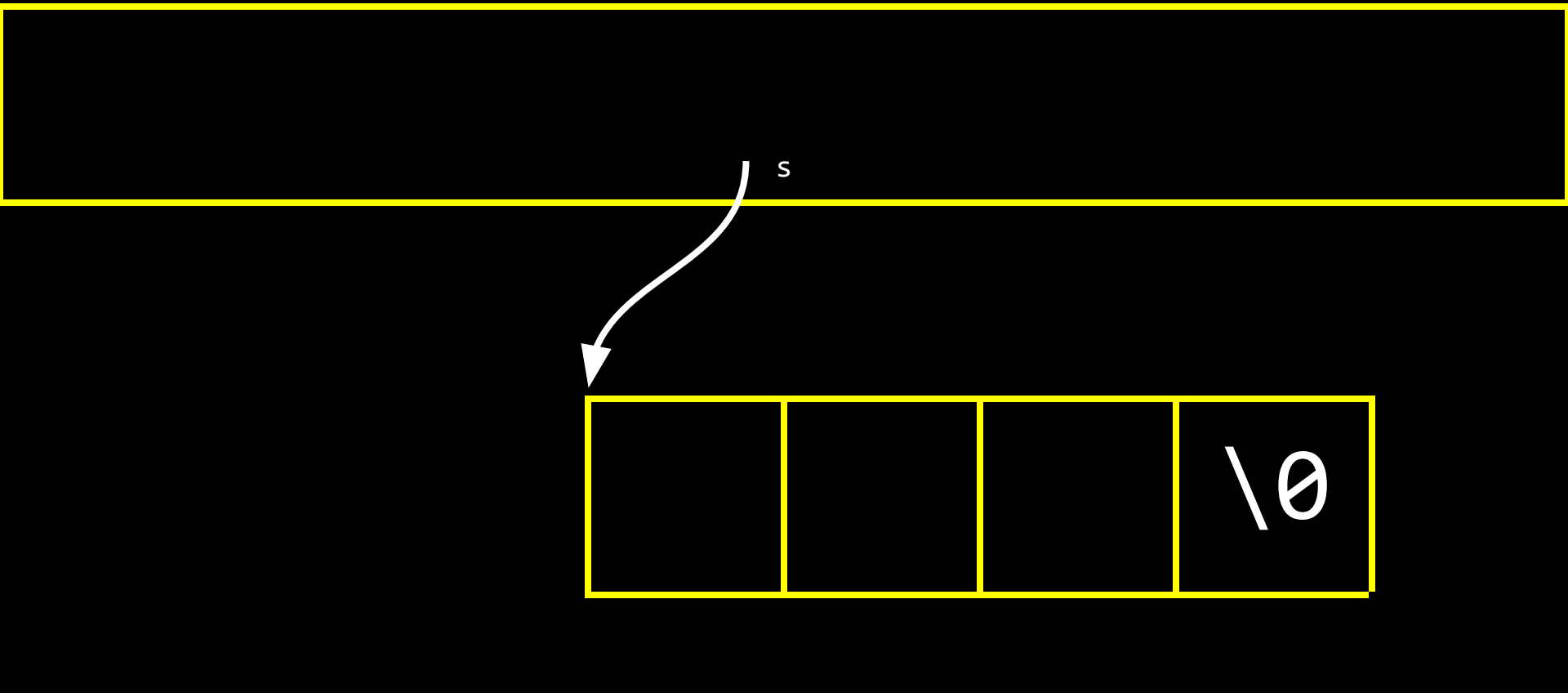

H
s[0]
I
s[1]
!
s[2]
\0
s[3]

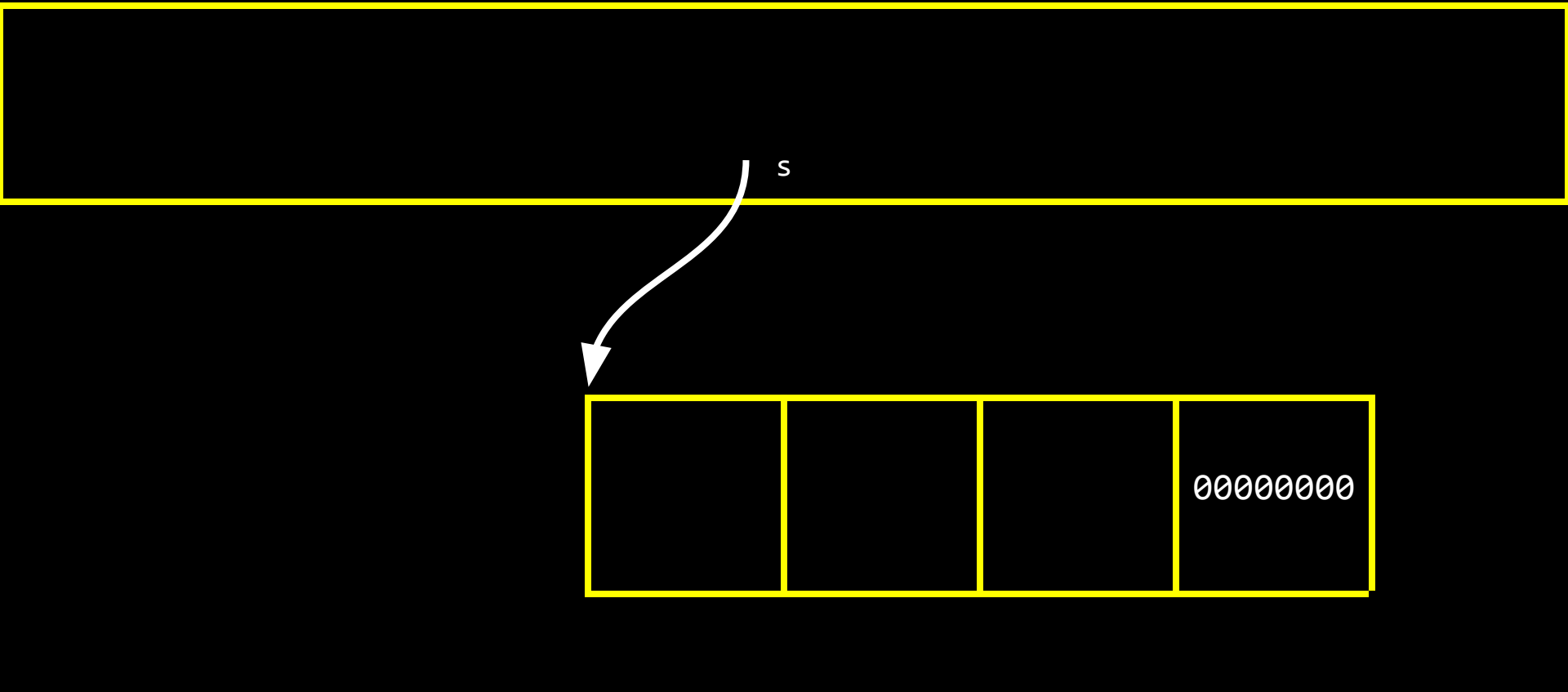

H
0x123
I
0x124
!
0x125
\0
0x126

0x123

s


s
H
I
!
\0


s
\0


s
00000000

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

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

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

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

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

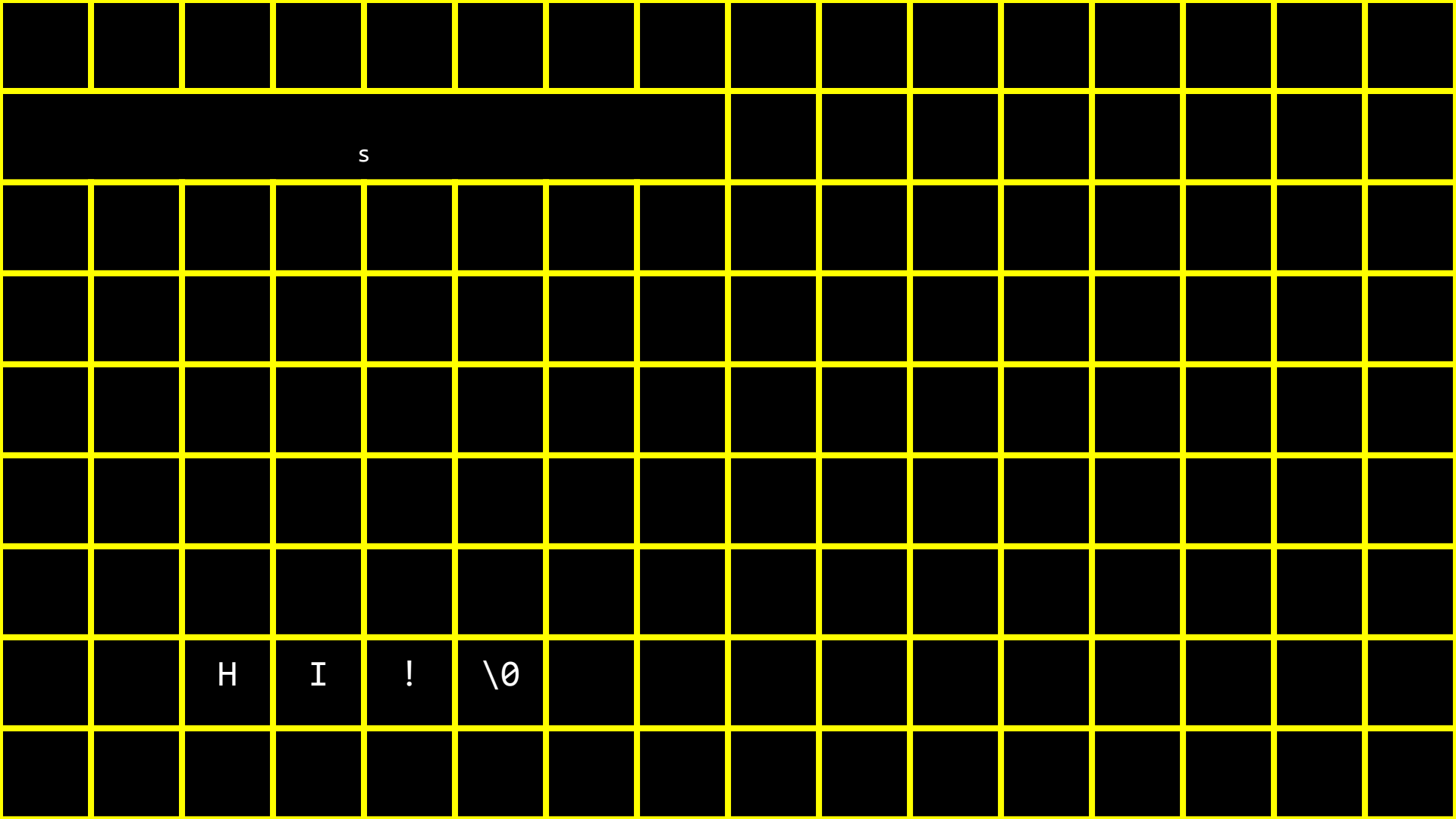
```
typedef char *string;
```

pointer arithmetic

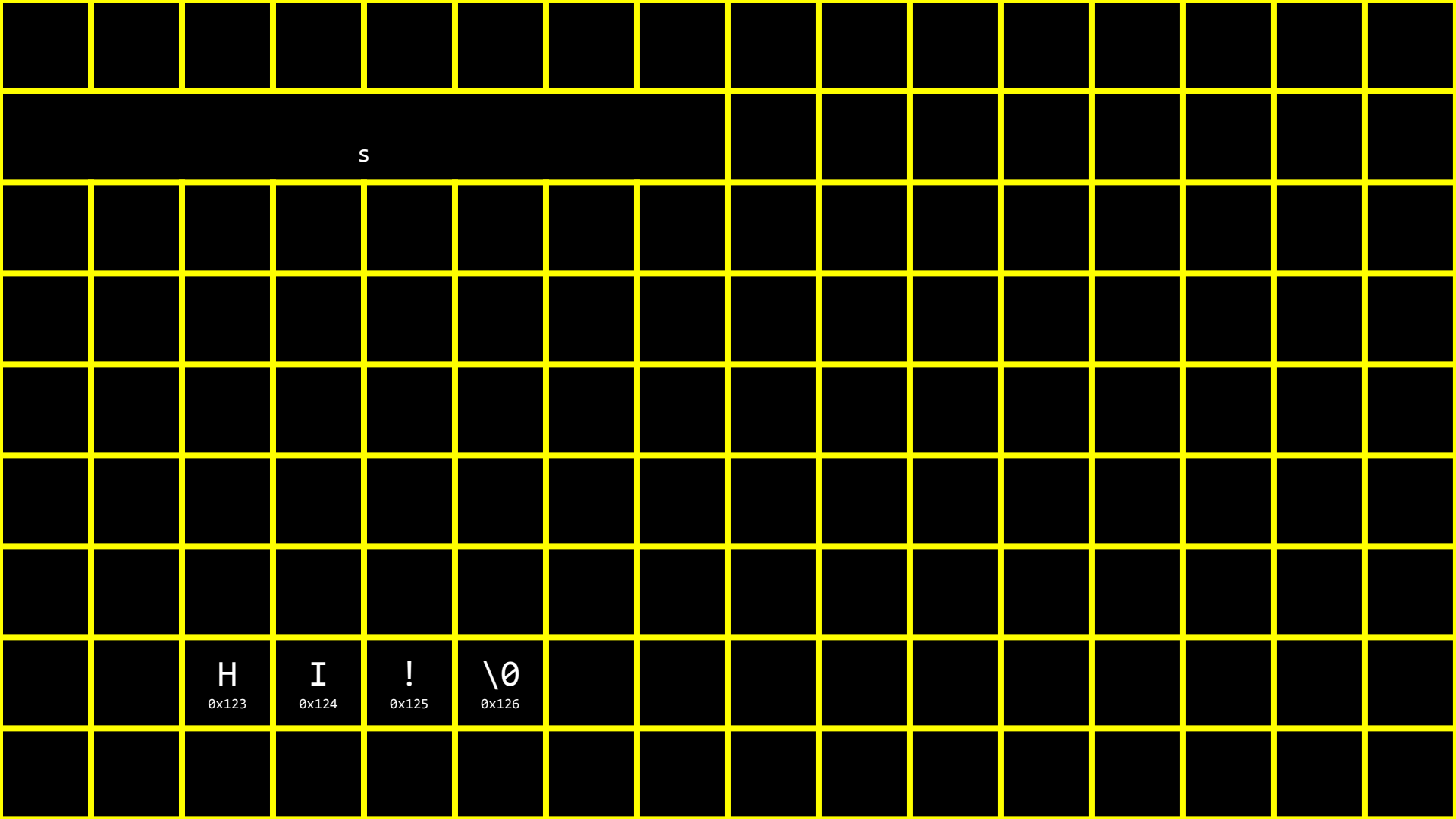
string

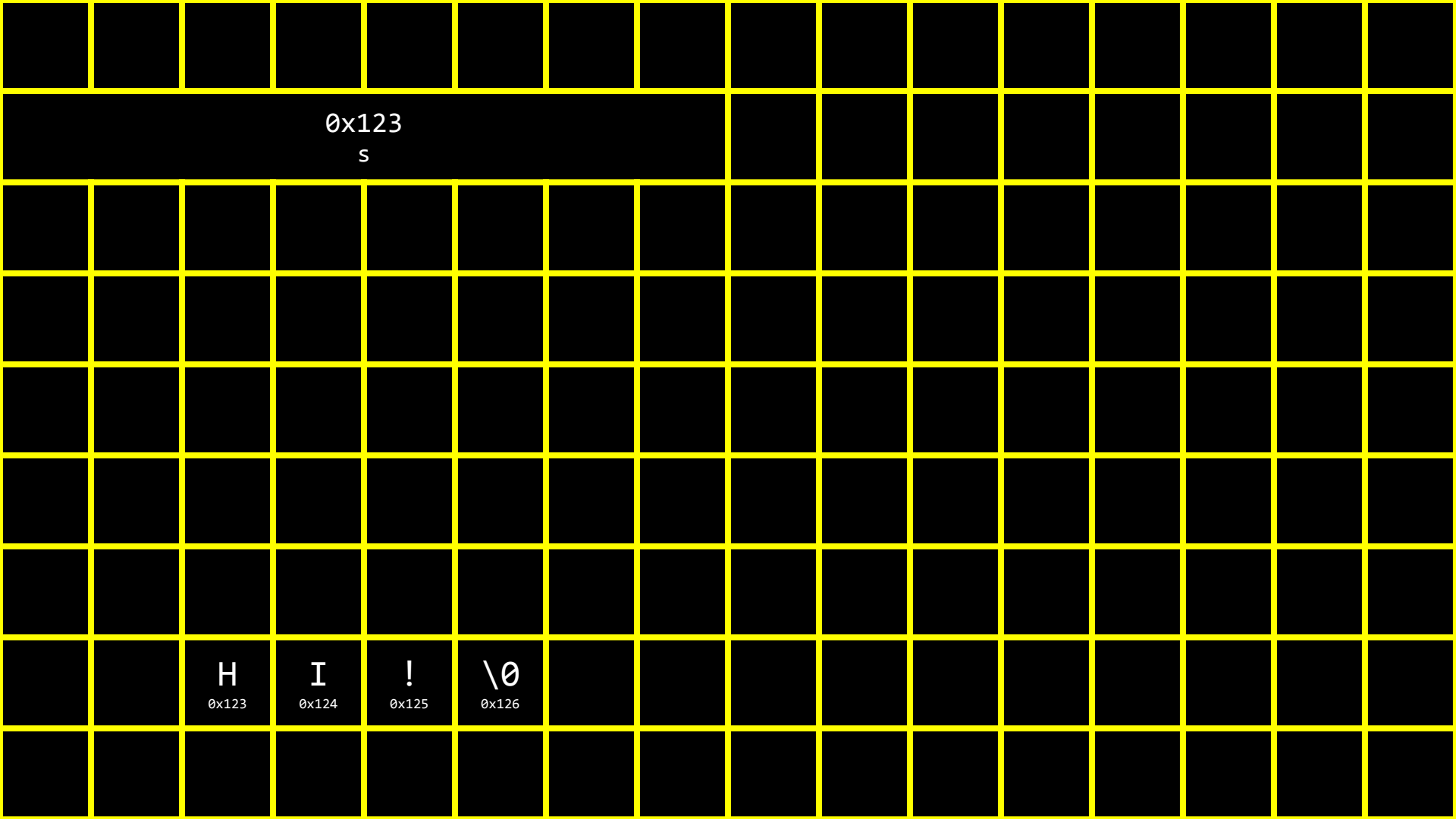
```
char *
```

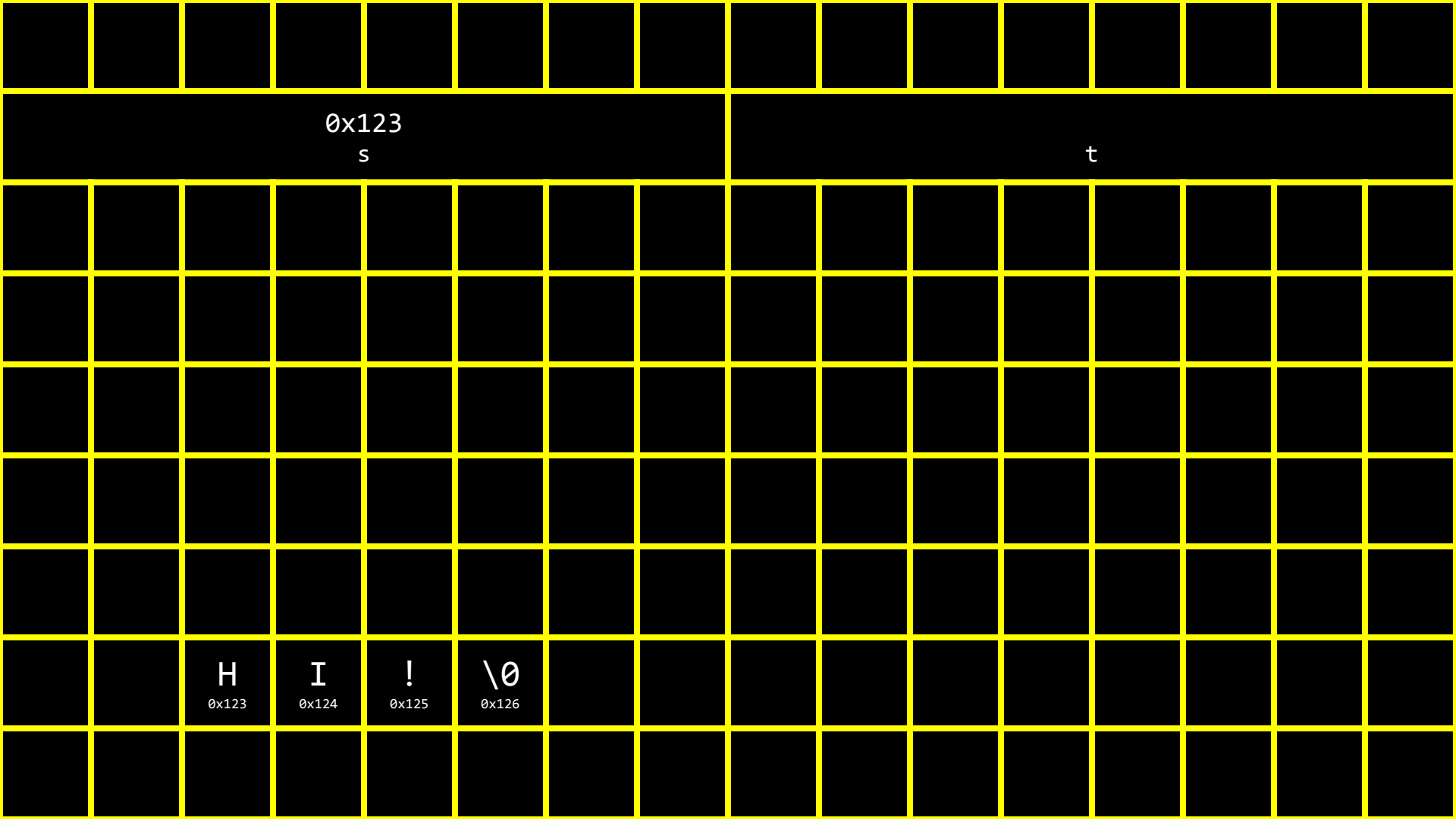
s

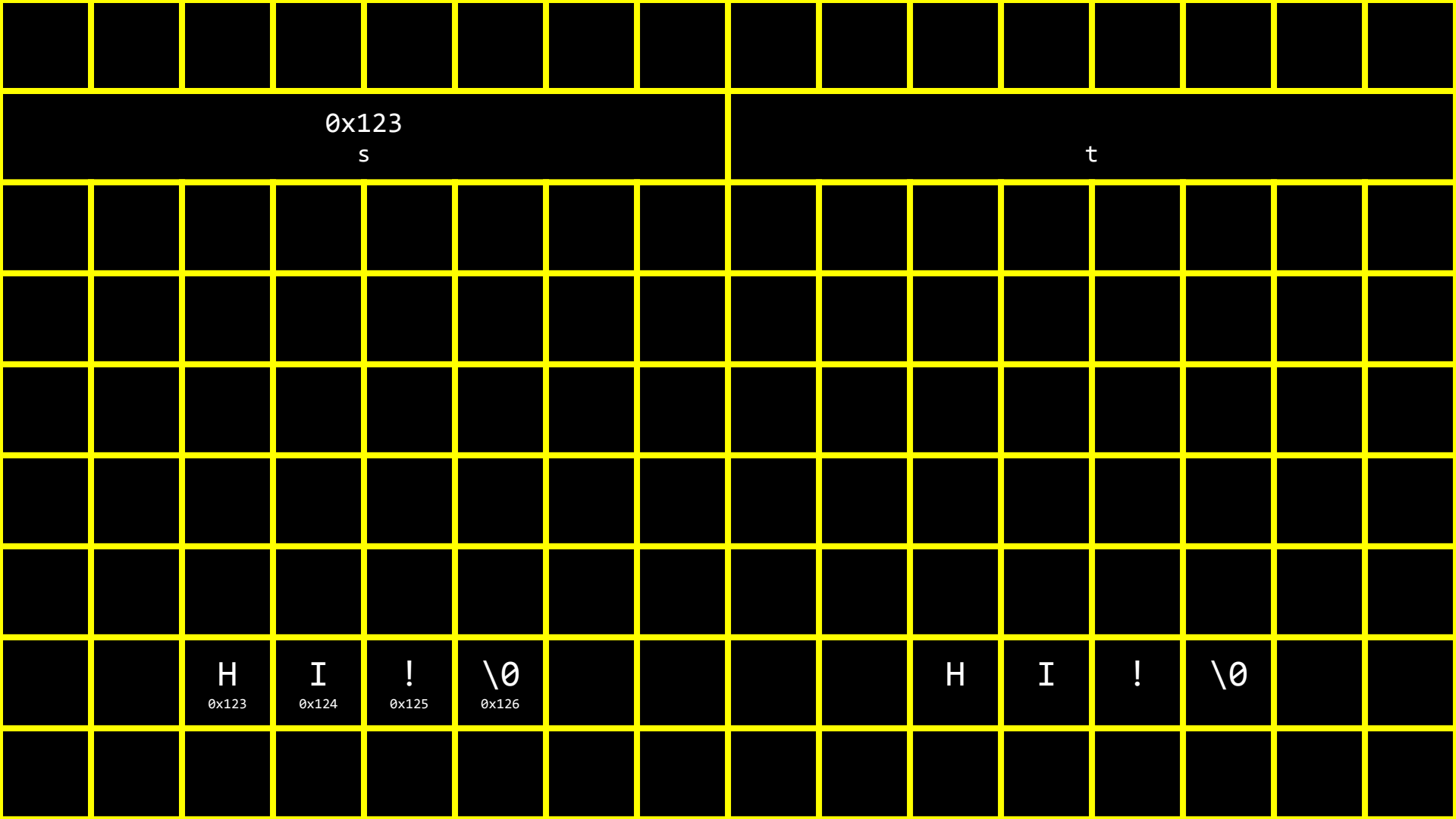

s
H
I
!
\0

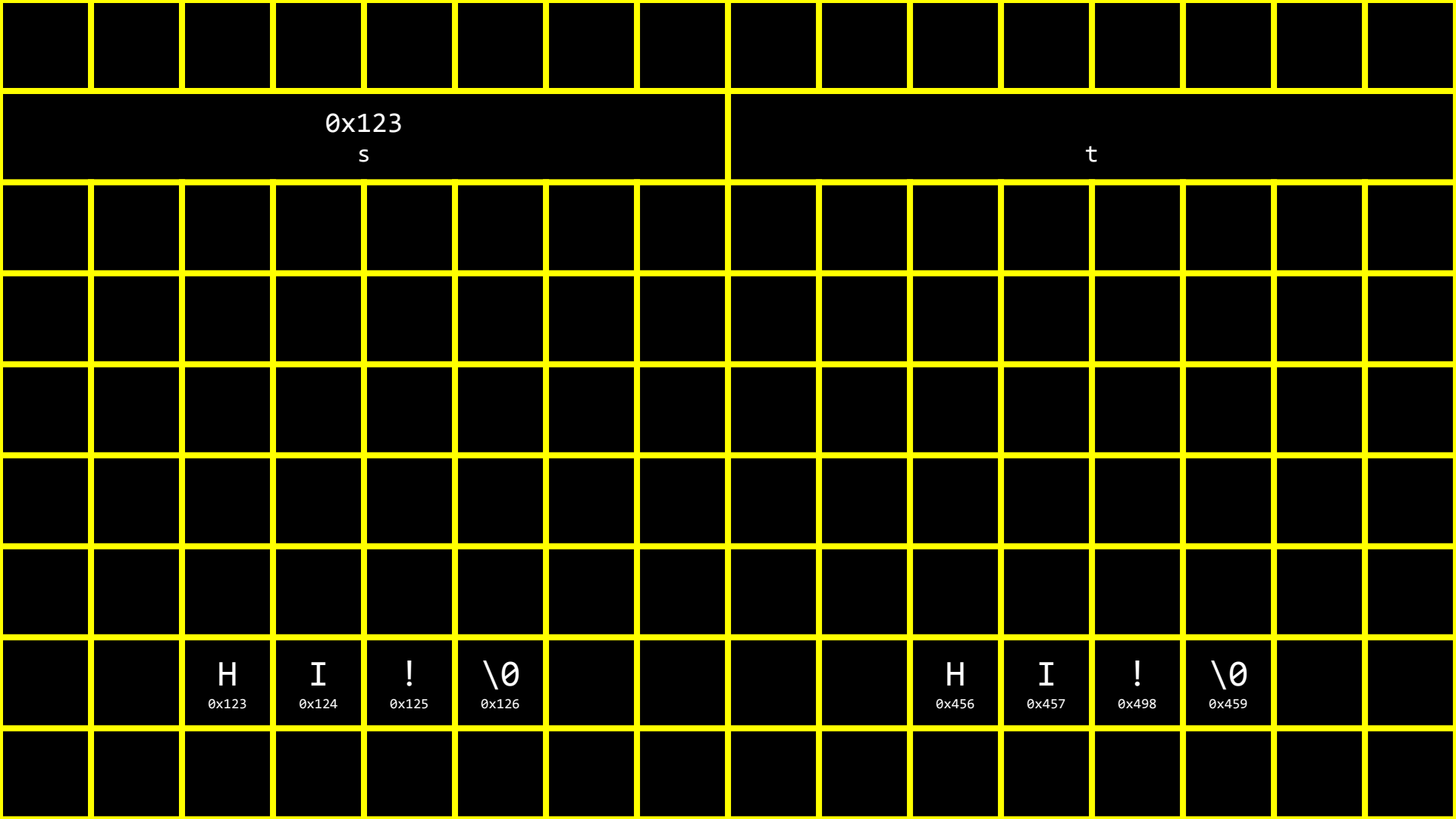
s

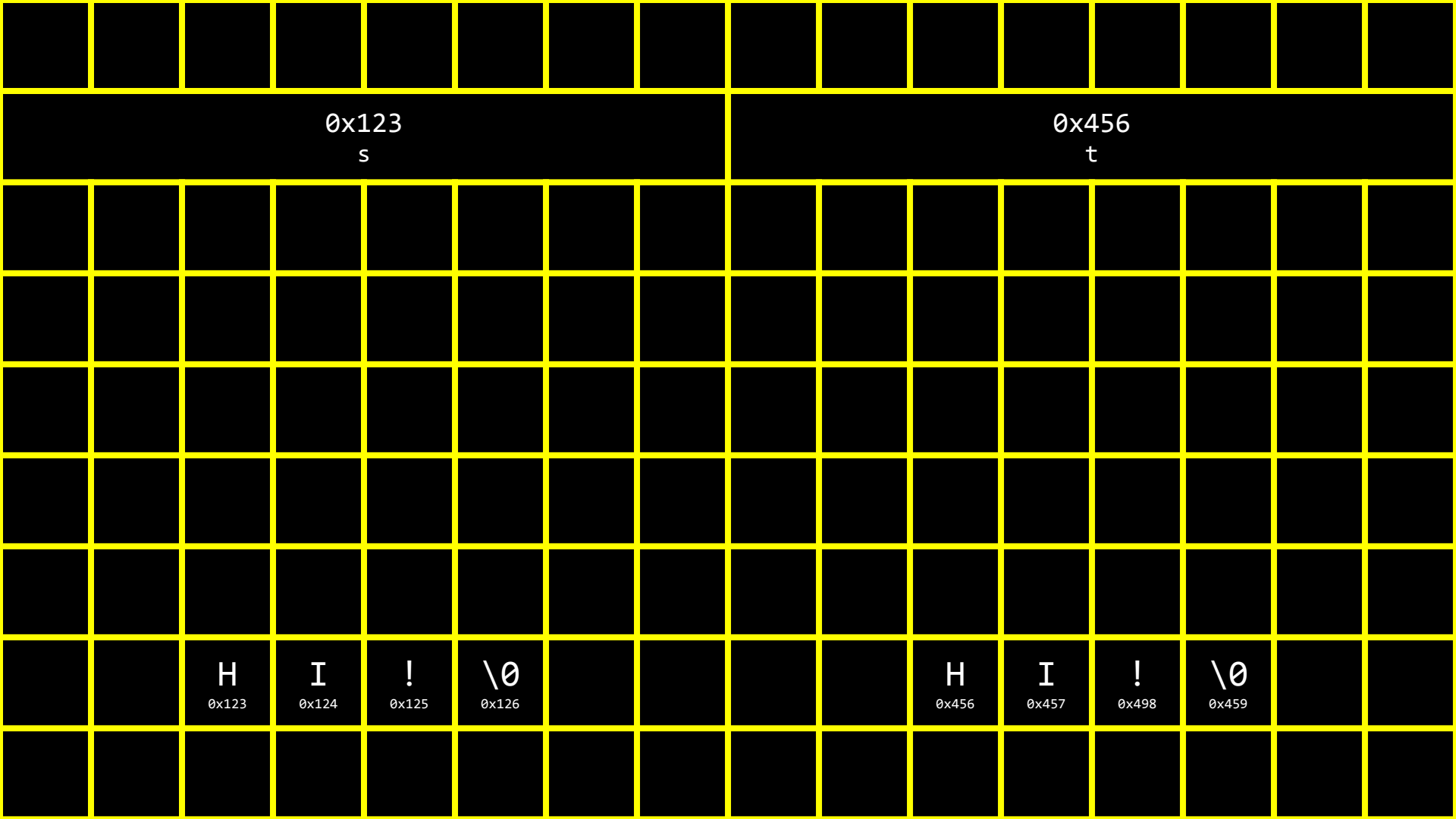


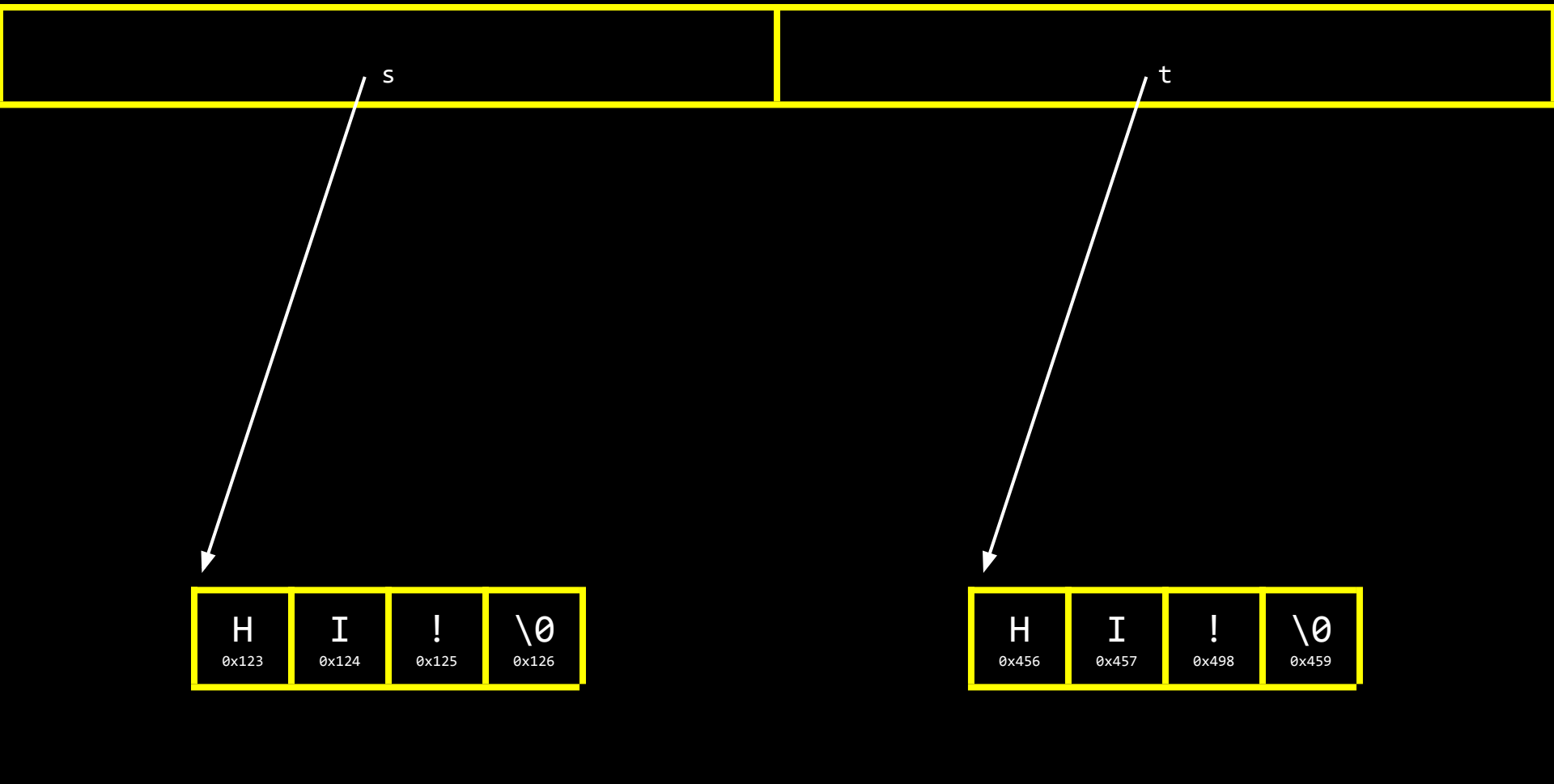

0x123
s
H
0x123
I
0x124
!
0x125
\0
0x126


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


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

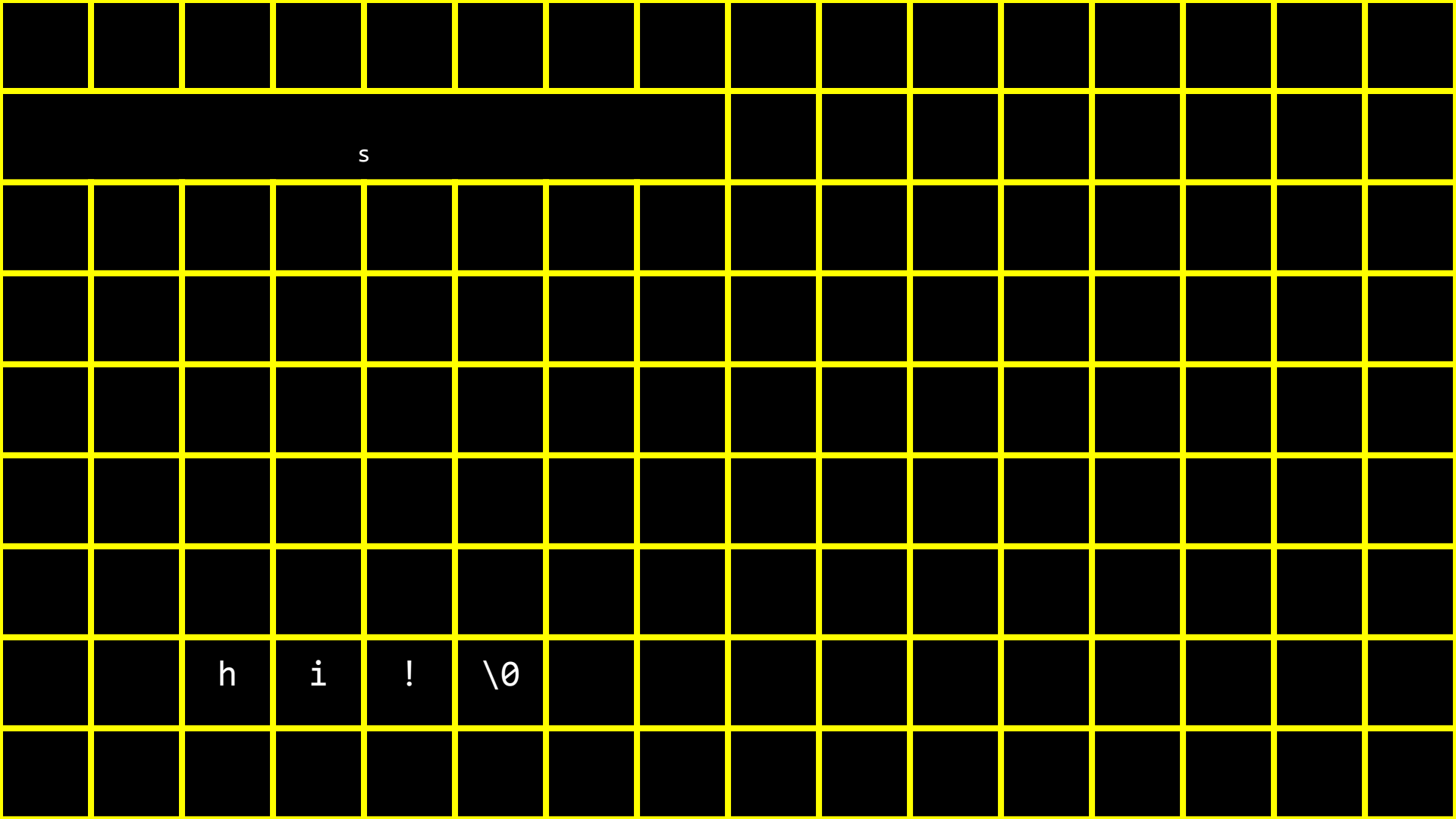

0x123
s
t
H
0x123
I
0x124
!
0x125
\0
0x126
H
0x456
I
0x457
!
0x498
\0
0x459


0x123
s
0x456
t
H
0x123
I
0x124
!
0x125
\0
0x126
H
0x456
I
0x457
!
0x498
\0
0x459


s
t
H
0x123
I
0x124
!
0x125
\0
0x126
H
0x456
I
0x457
!
0x498
\0
0x459

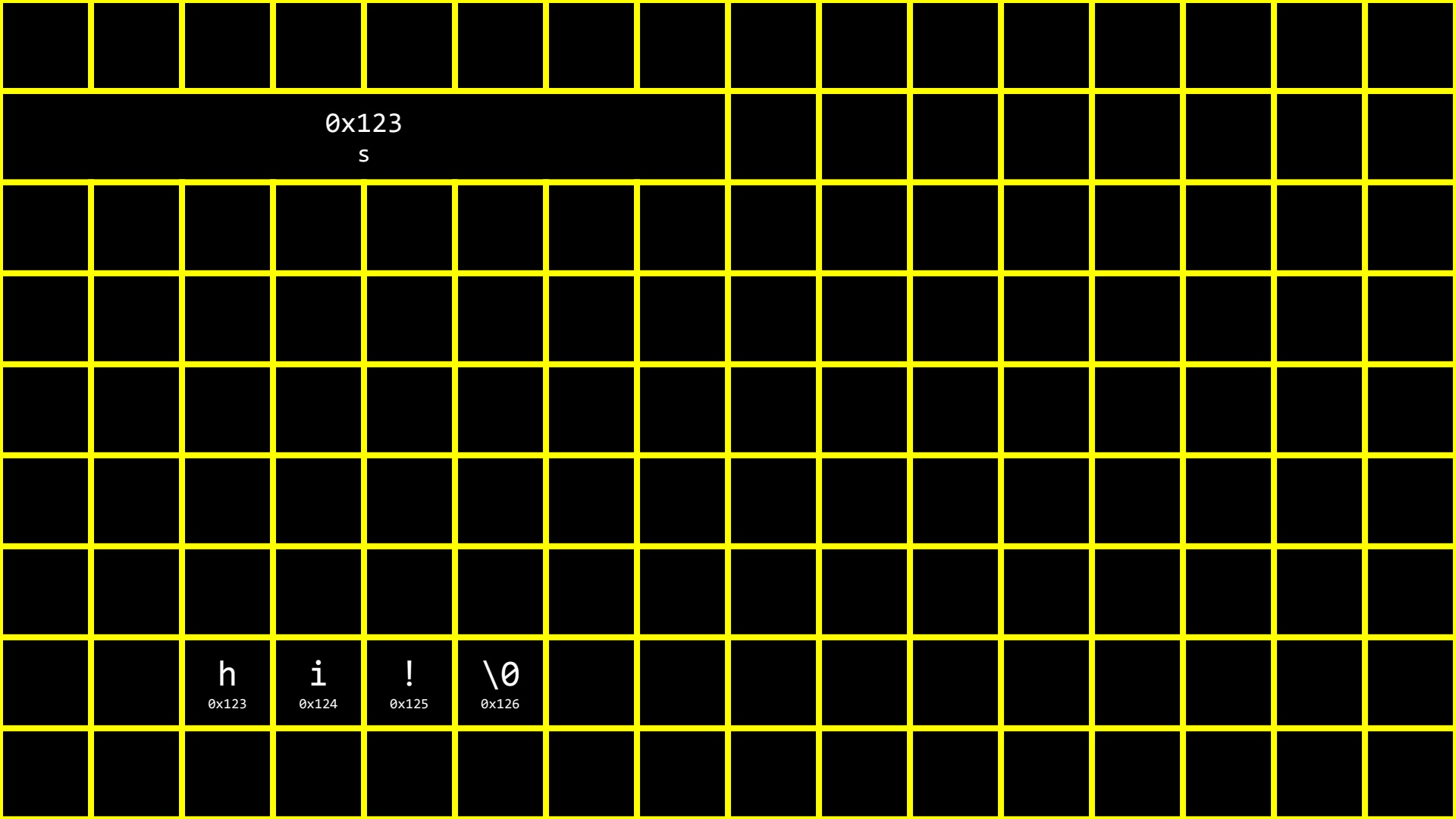
```
char *
```

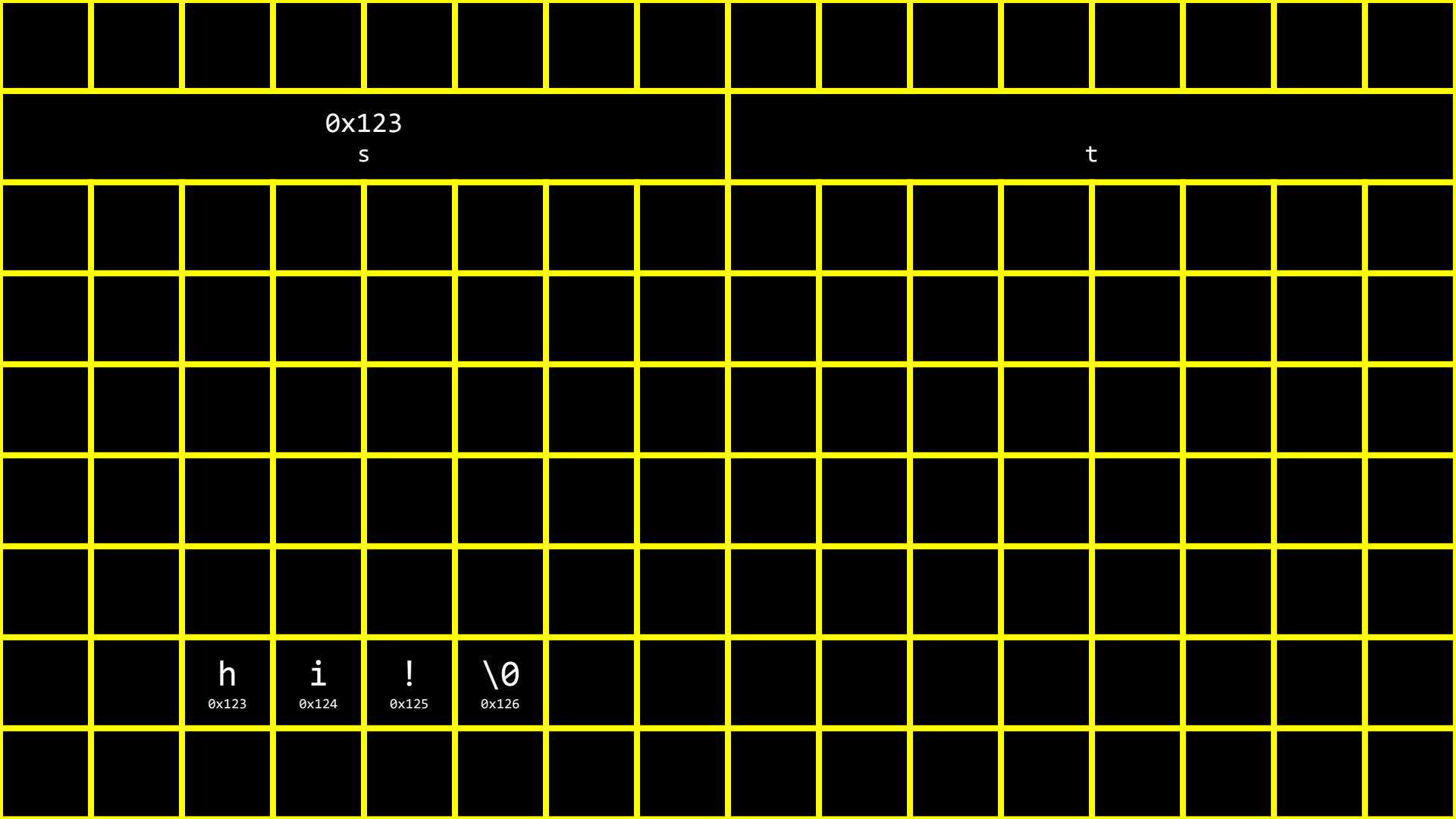
s

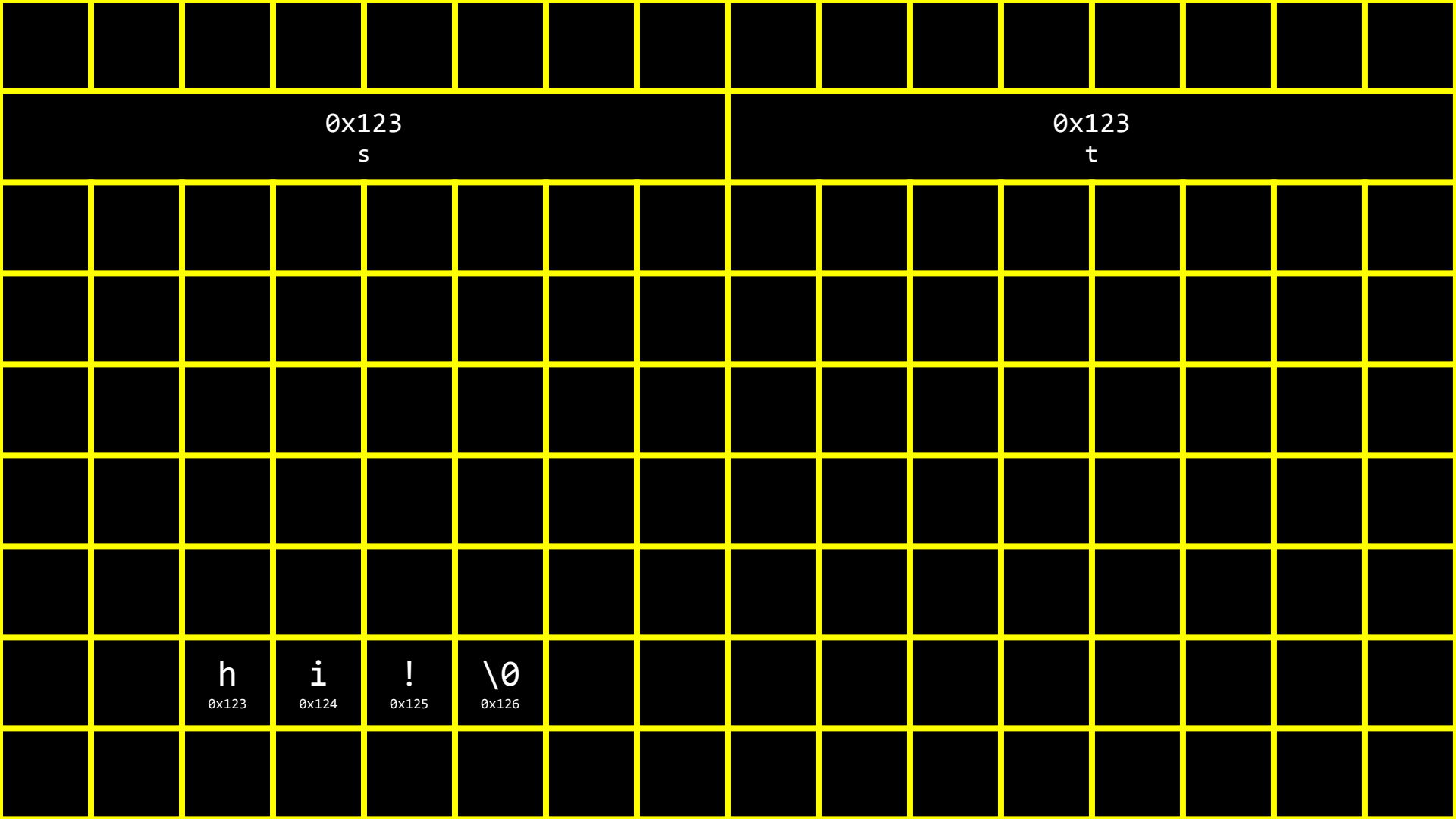

s
h
i
!
\0

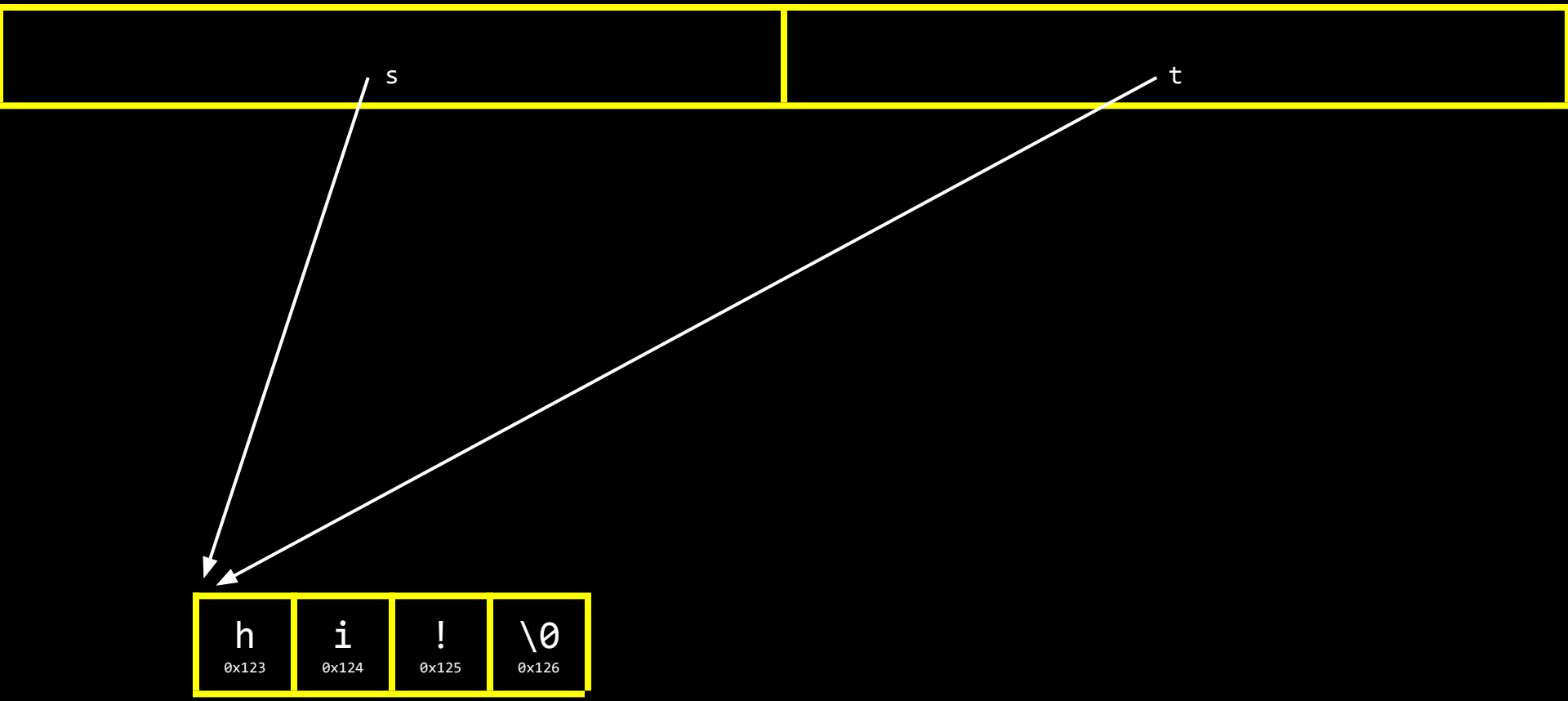
s

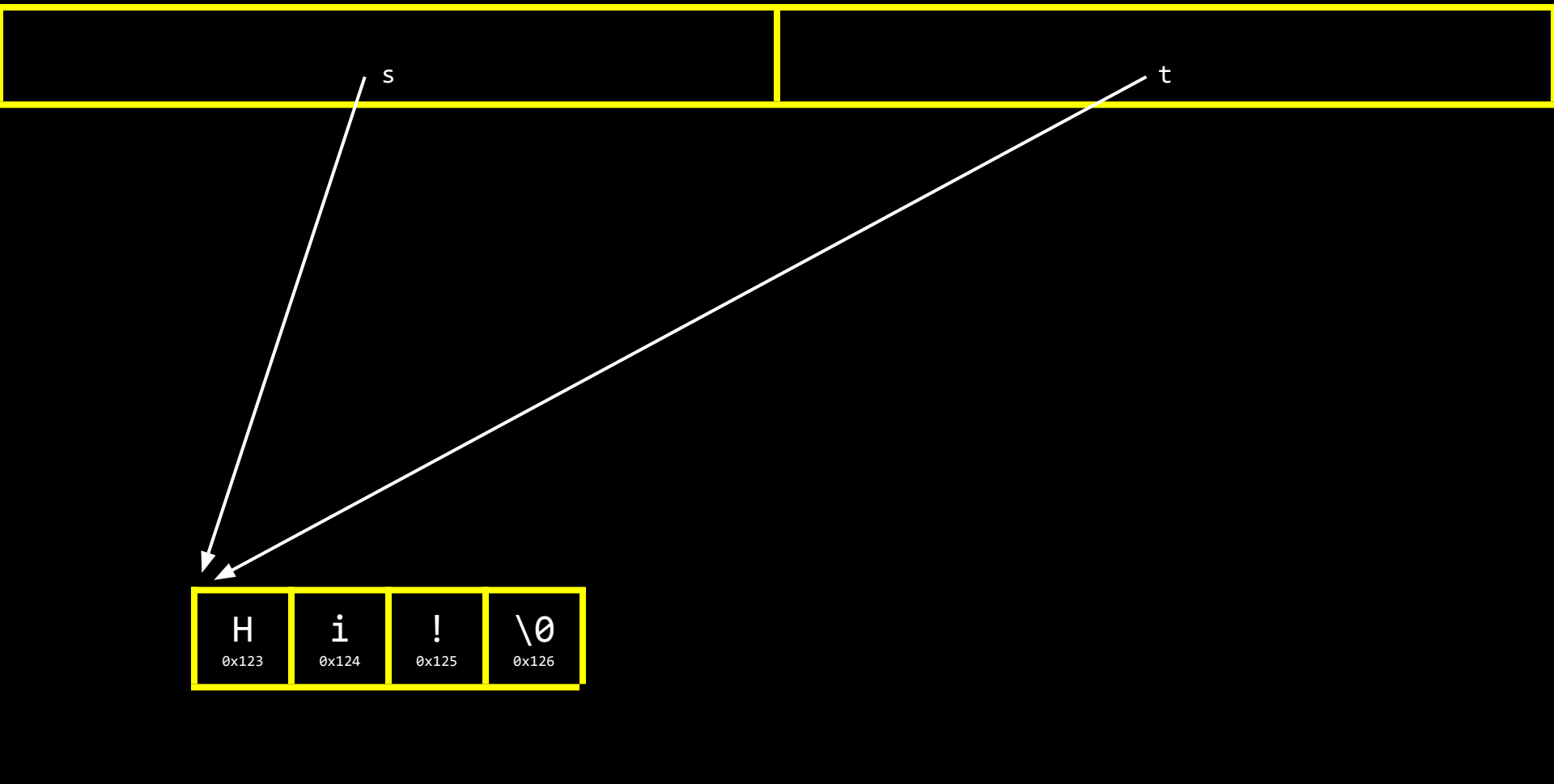
h	i	!	\0
0x123	0x124	0x125	0x126


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


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


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

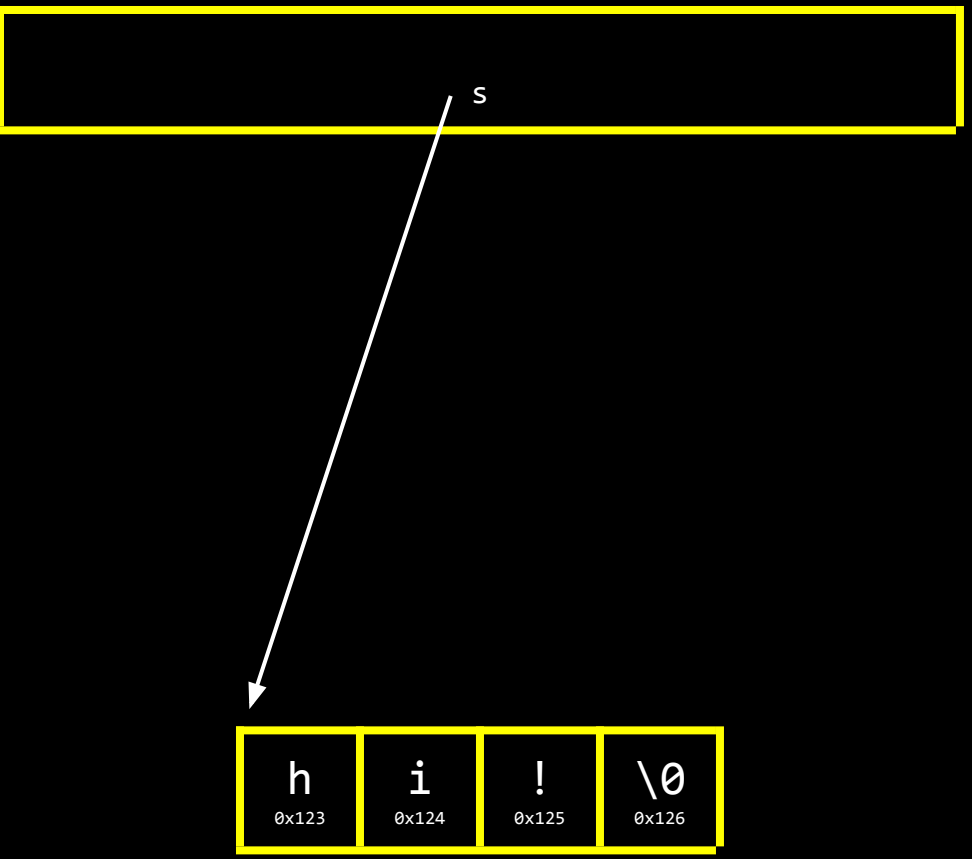

s
t
h
0x123
i
0x124
!
0x125
\0
0x126


s
t
H
0x123
i
0x124
!
0x125
\0
0x126

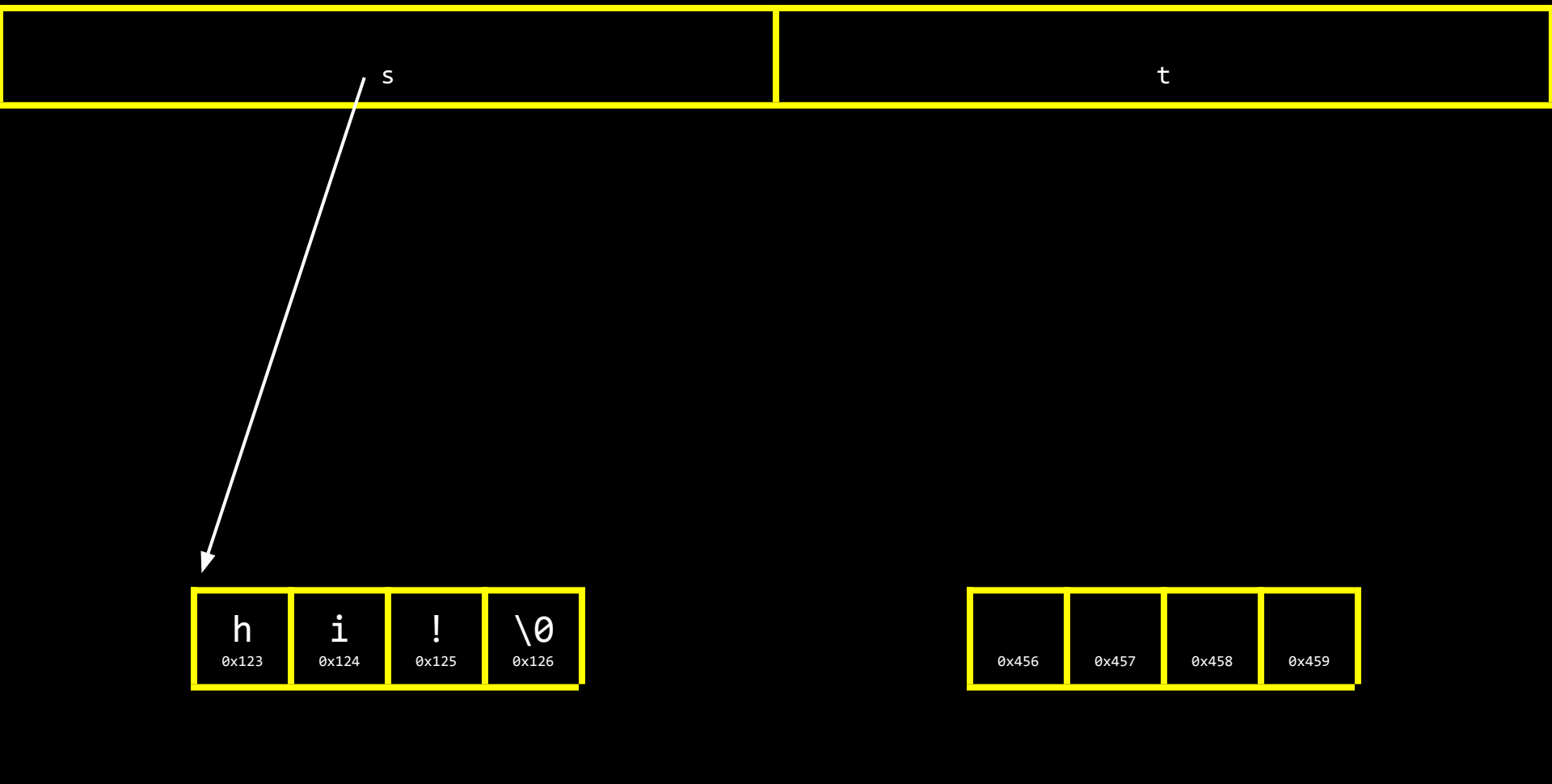
malloc

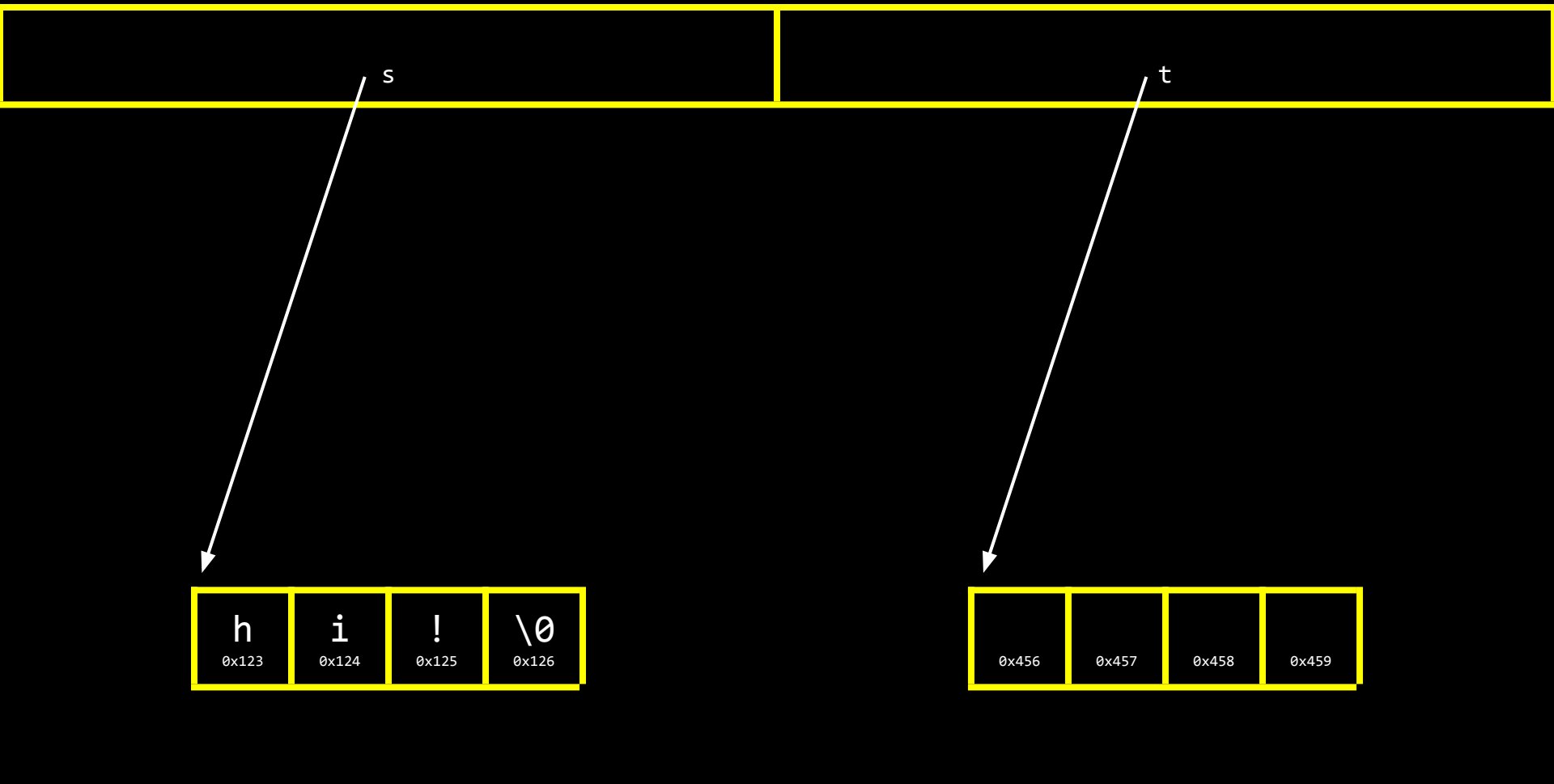
free

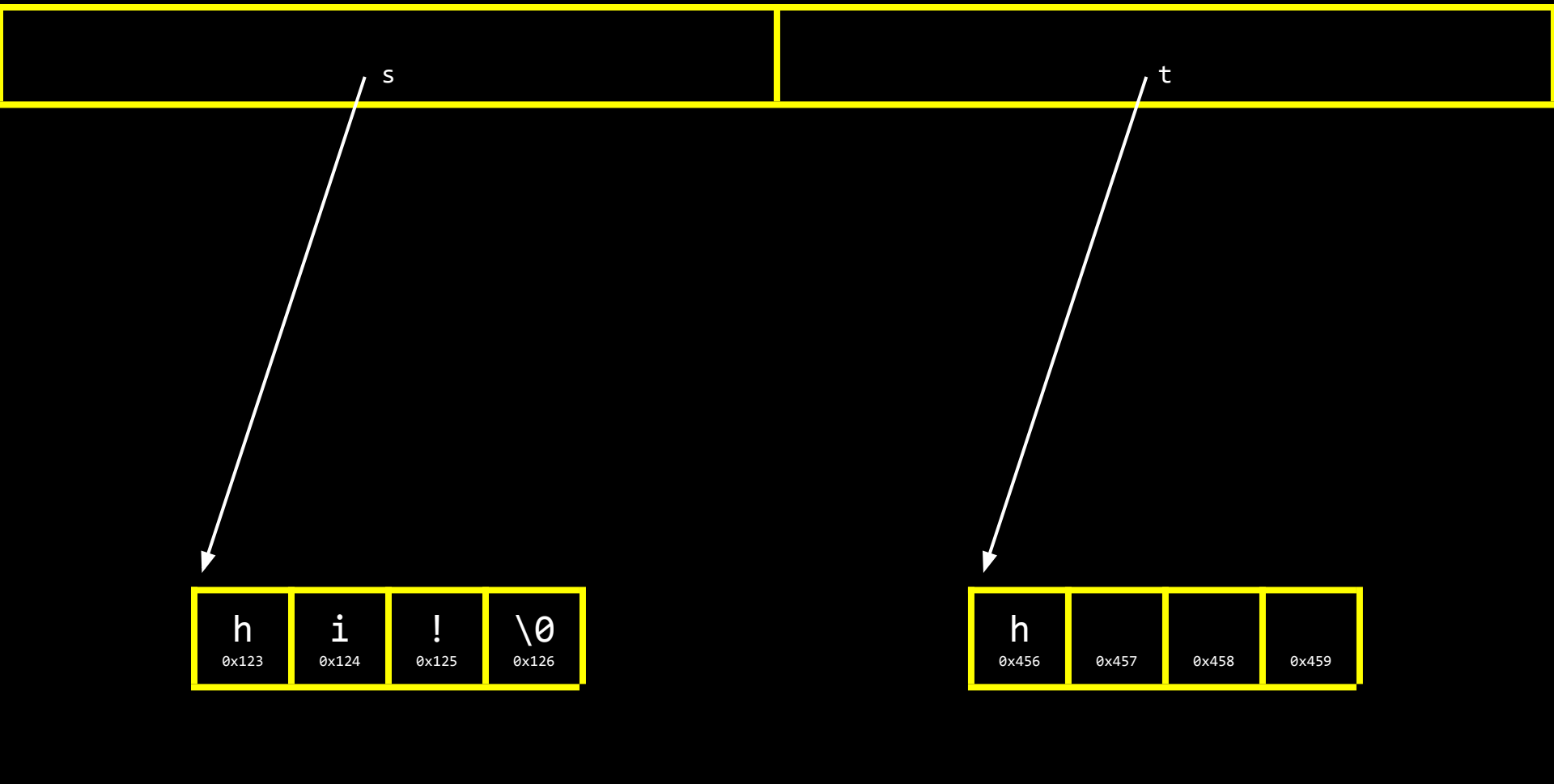
…

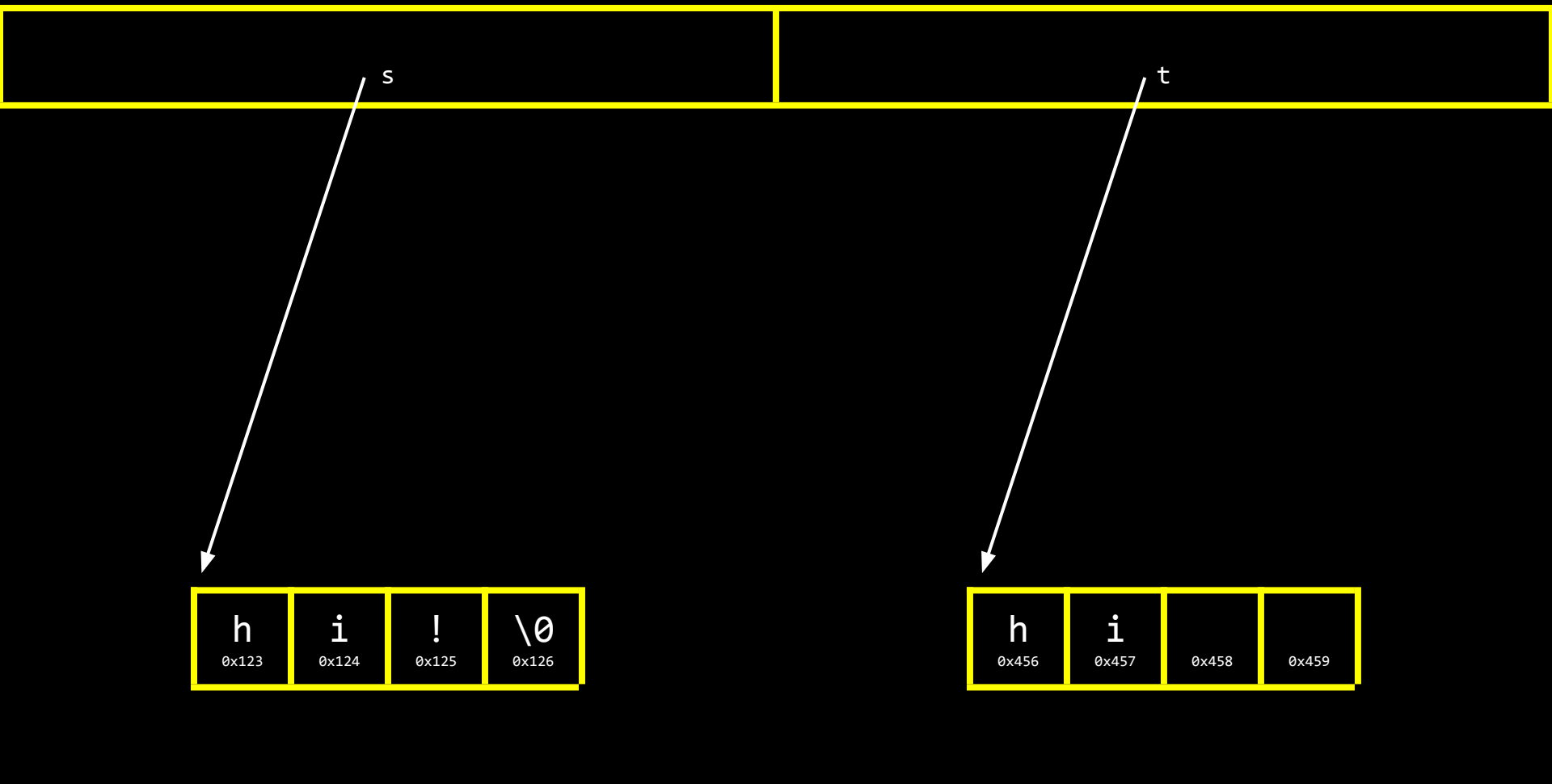

s
h
0x123
i
0x124
!
0x125
\0
0x126

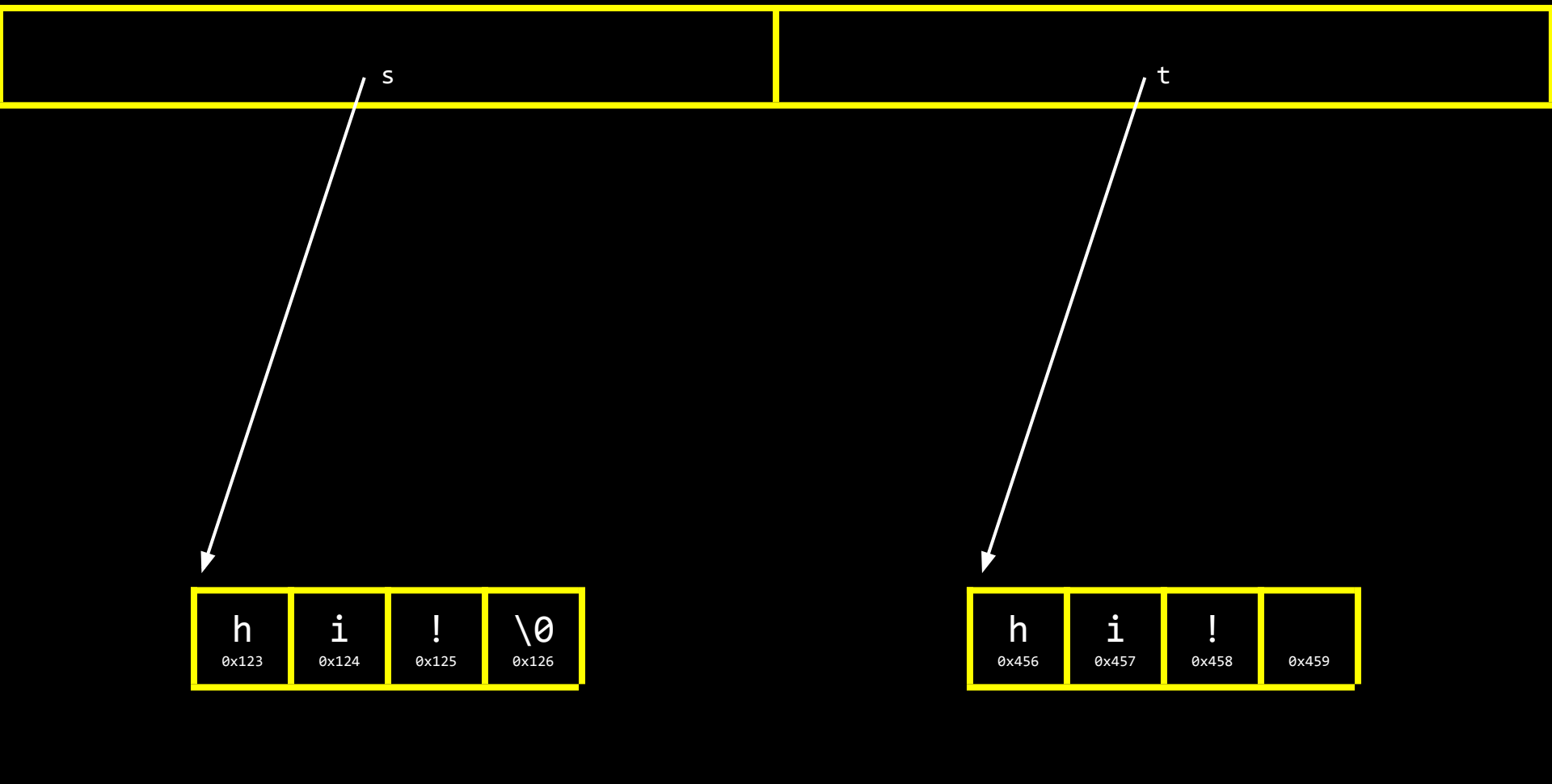

s
t
h
0x123
i
0x124
!
0x125
\0
0x126

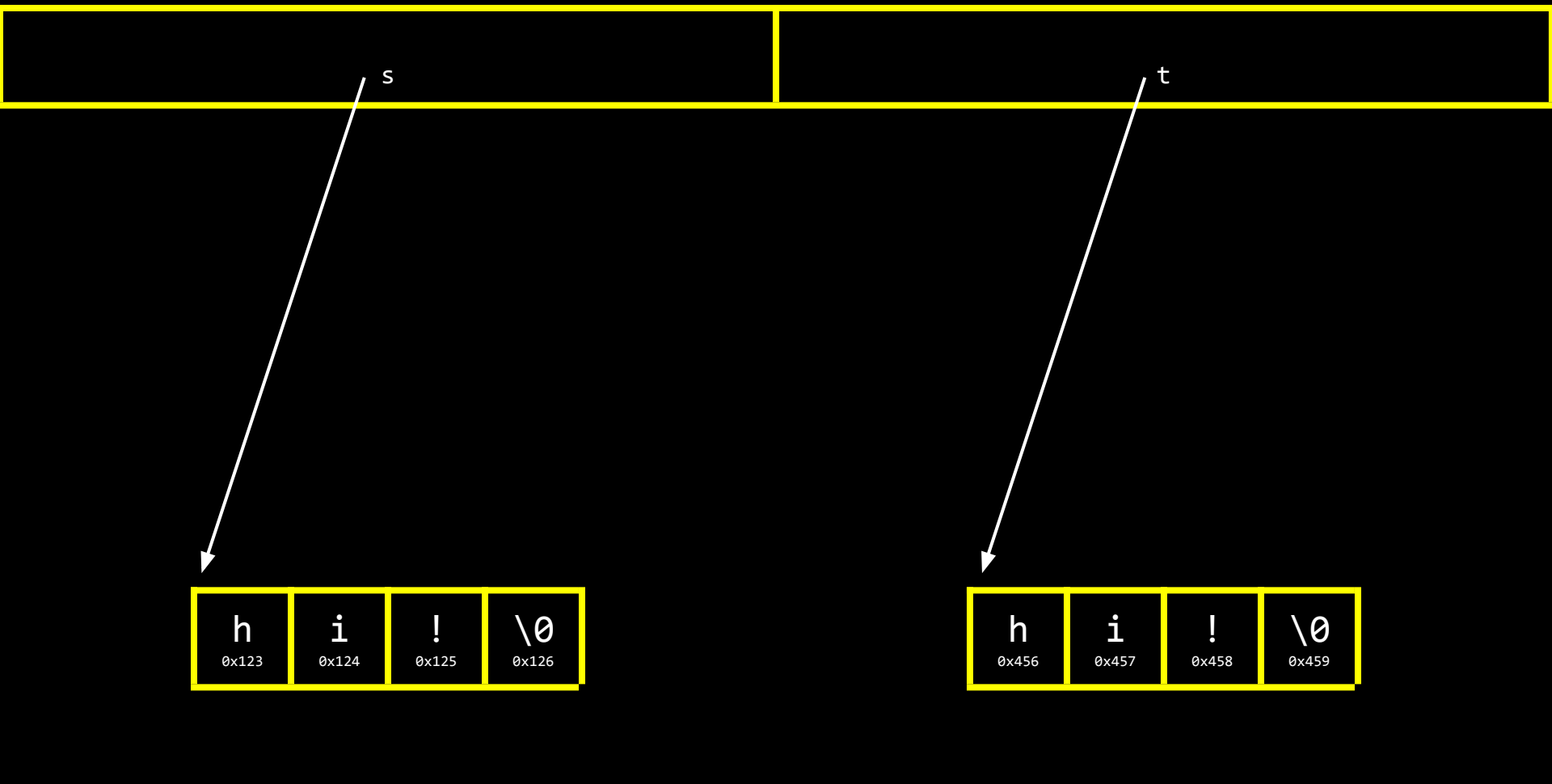

s
t
h
0x123
i
0x124
!
0x125
\0
0x126
0x456
0x457
0x458
0x459

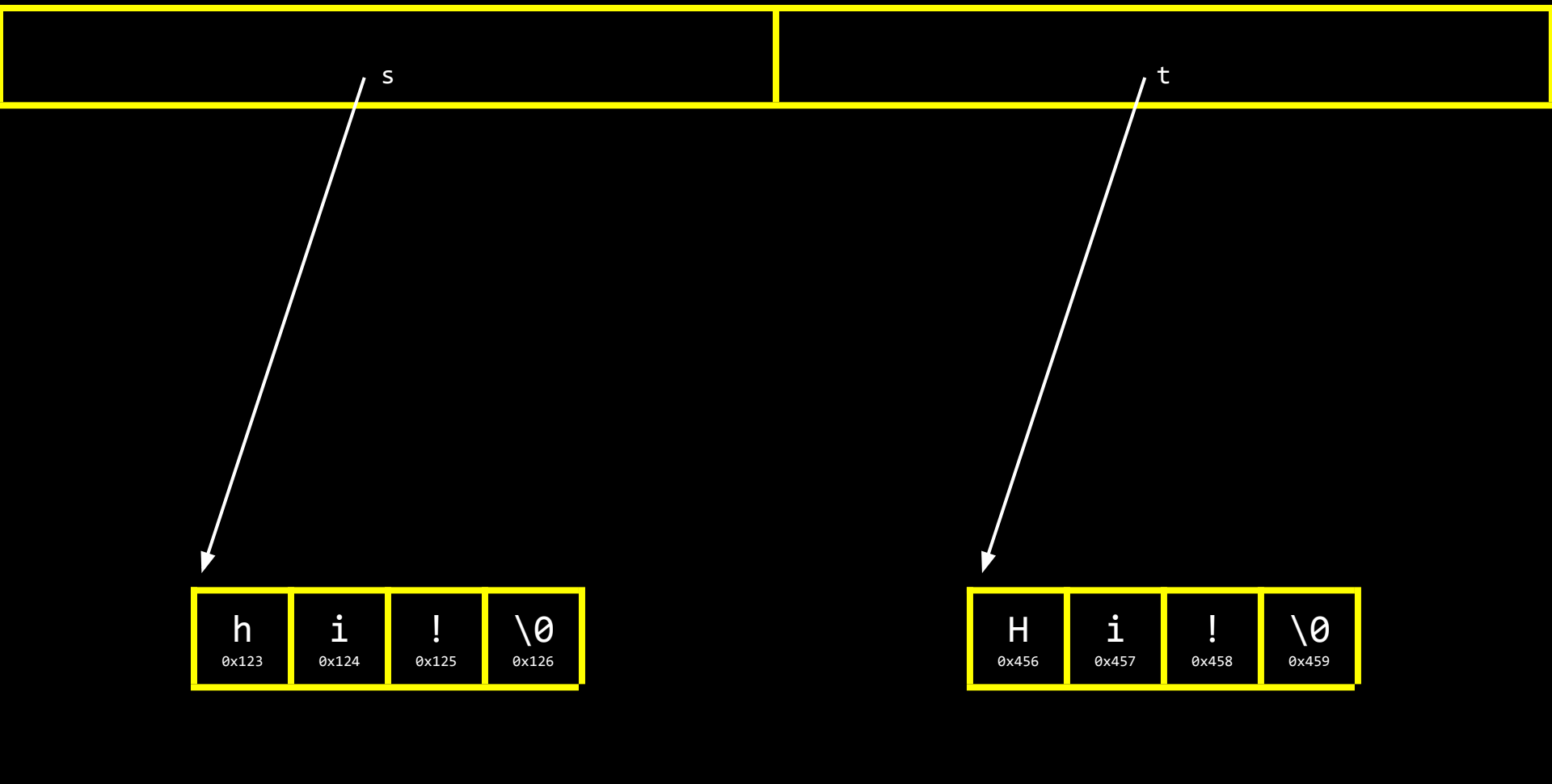

s
t
h
0x123
i
0x124
!
0x125
\0
0x126
0x456
0x457
0x458
0x459


s
t
h
0x123
i
0x124
!
0x125
\0
0x126
h
0x456
0x457
0x458
0x459


s
t
h
0x123
i
0x124
!
0x125
\0
0x126
h
0x456
i
0x457
0x458
0x459


s
t
h
0x123
i
0x124
!
0x125
\0
0x126
h
0x456
i
0x457
!
0x458
0x459


s
t
h
0x123
i
0x124
!
0x125
\0
0x126
h
0x456
i
0x457
!
0x458
\0
0x459


s
t
h
0x123
i
0x124
!
0x125
\0
0x126
H
0x456
i
0x457
!
0x458
\0
0x459

valgrind

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

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

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

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

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

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

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

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

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

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

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

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

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

    x = malloc(sizeof(int));

    *x = 42;
    *y = 13;

    y = x;

    *y = 13;
}
```

garbage values

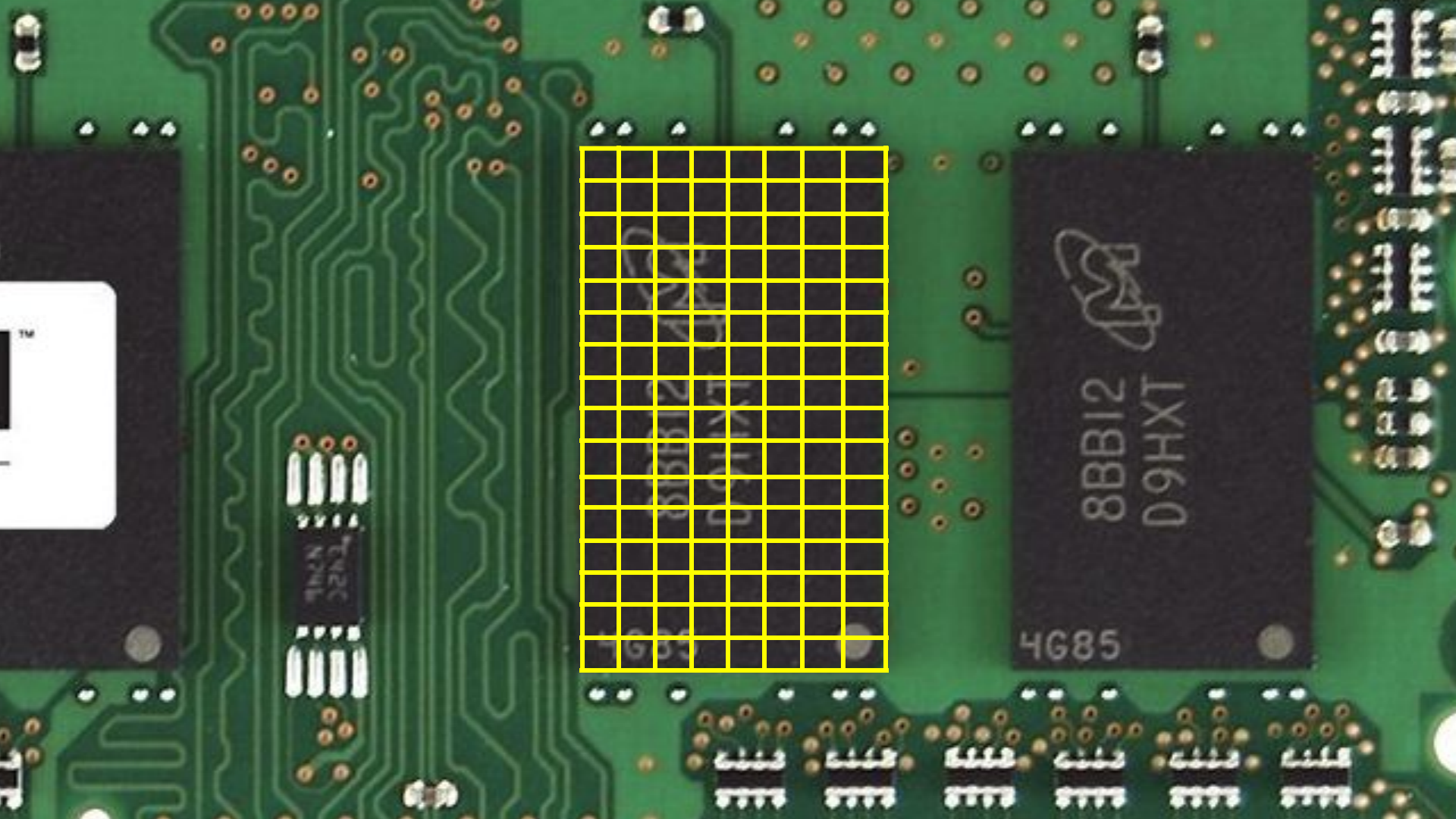
```
void swap(int a, int b)
{



}
```

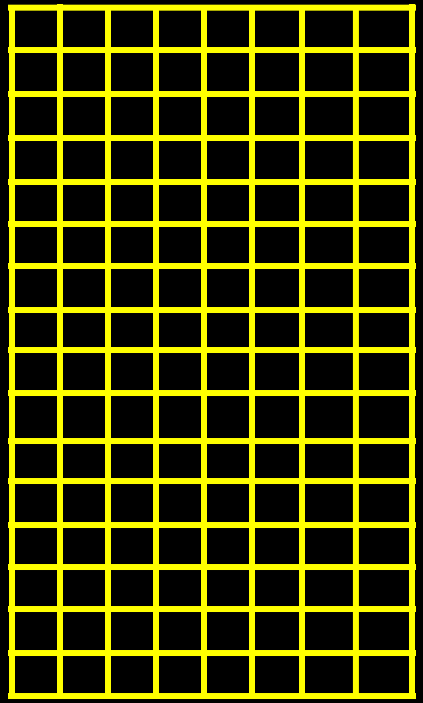
```
void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

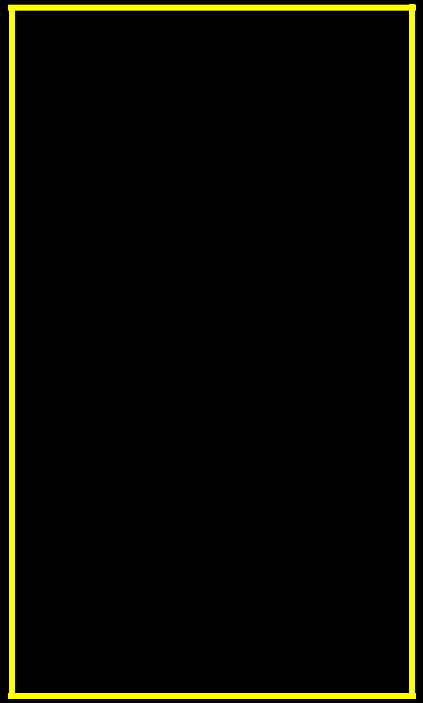
```
void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

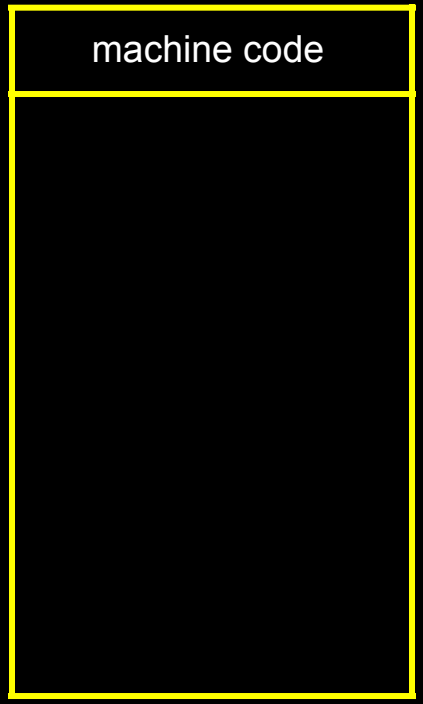

crucial
by Micron
8BBI2
D9HXT
8BBI2
D9HXT
4G85
4G85
4G85
47J6

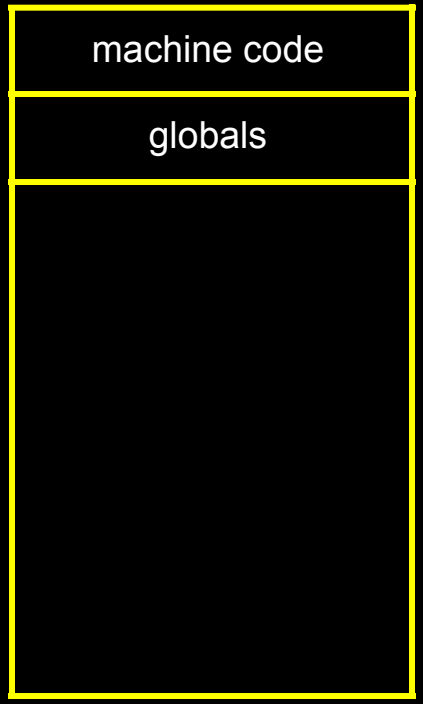

8BBI2
D9HXT
4G85
8BBI2
D9HXT
4G85

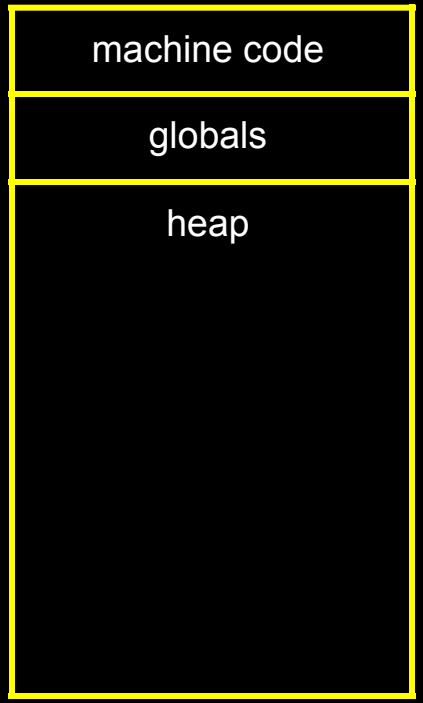

8BBI2
D9HXT
4G85

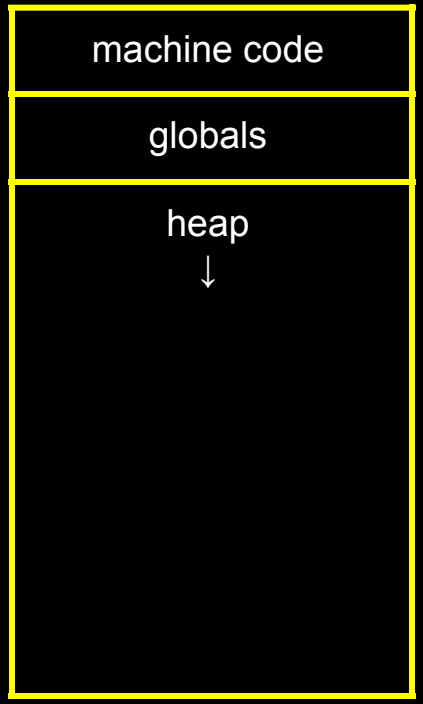


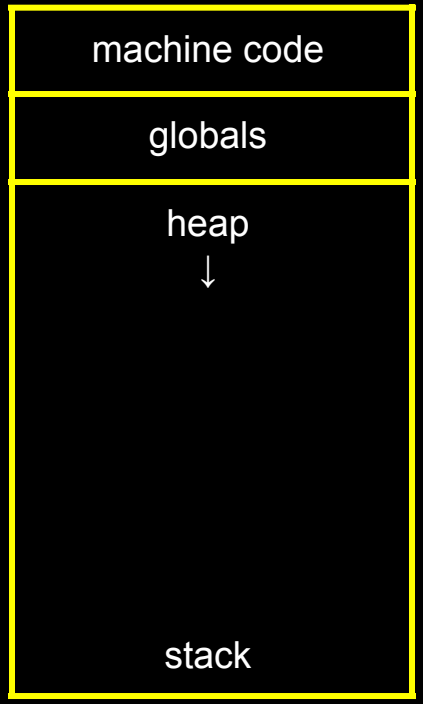


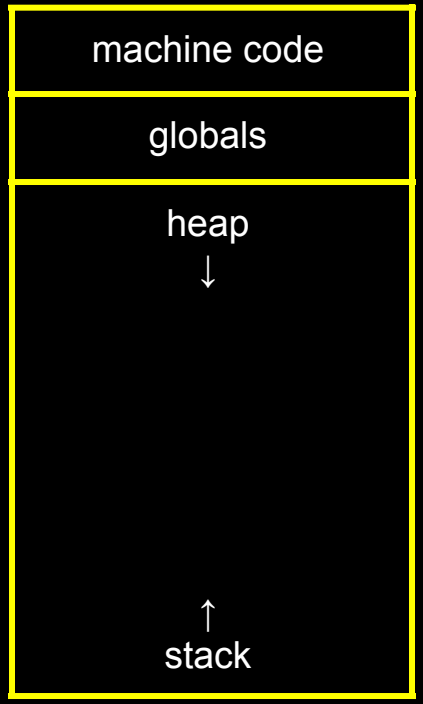

machine code

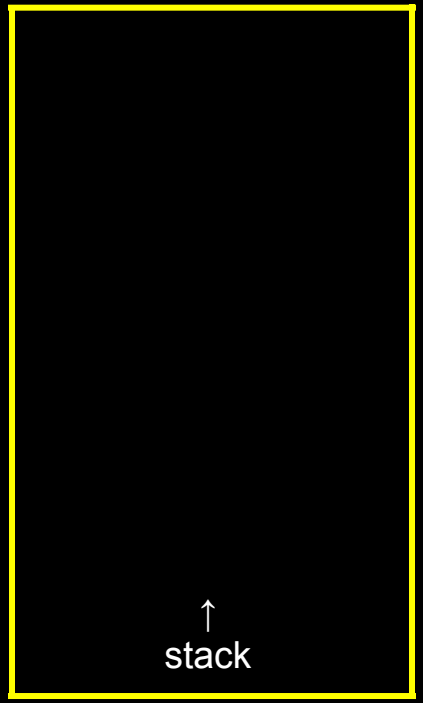

machine code
globals


machine code
globals
heap

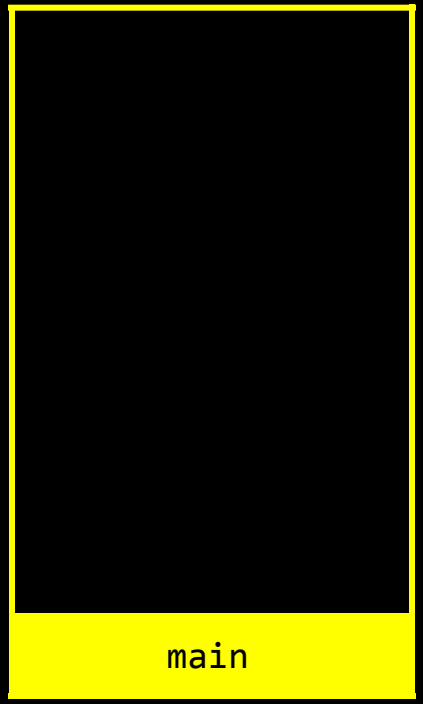

machine code
globals
heap
↓

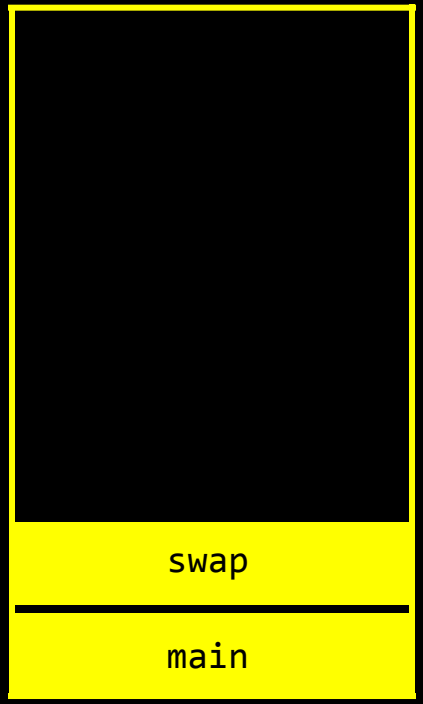

machine code
globals
heap
↓
stack

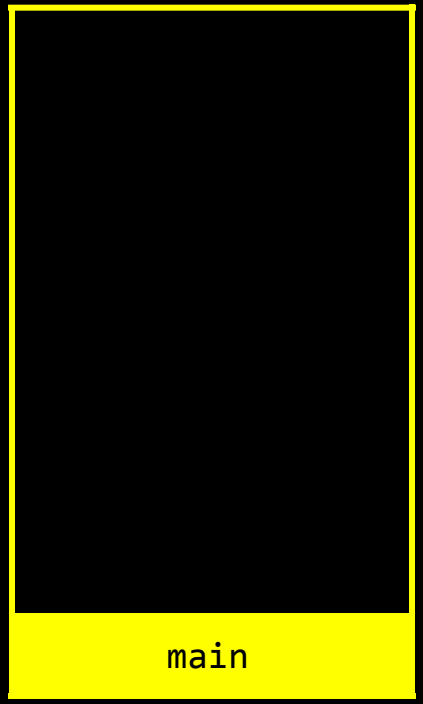

machine code
globals
heap
↓
↑
stack

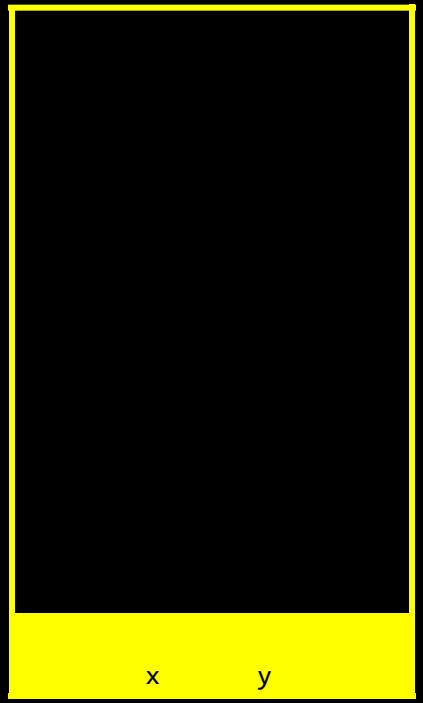

↑
stack

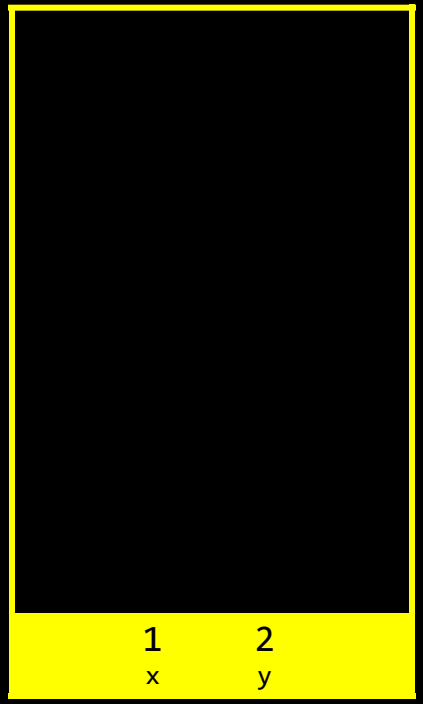
```
void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

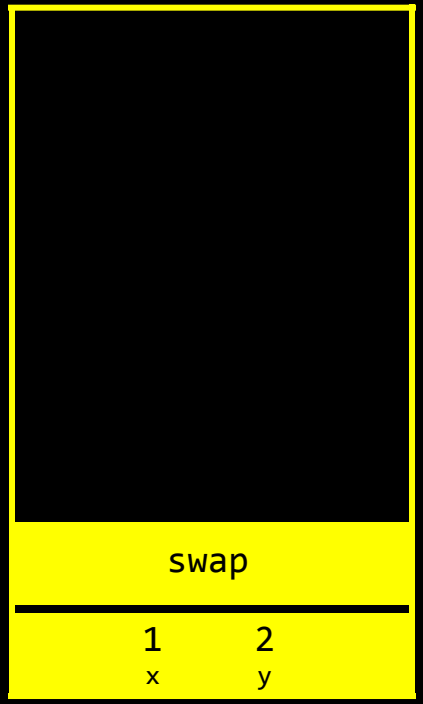

main

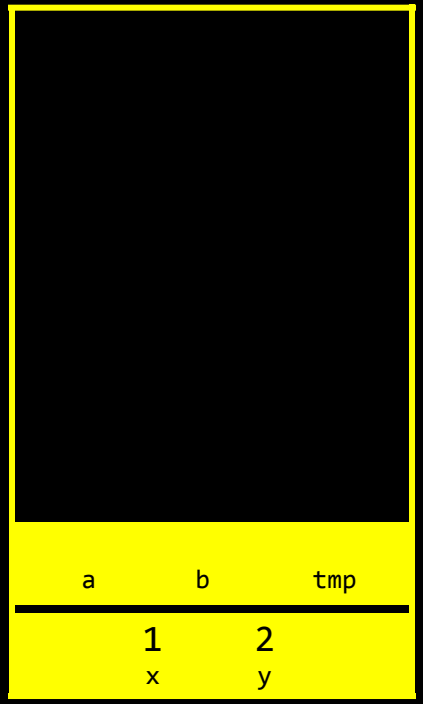

swap
main

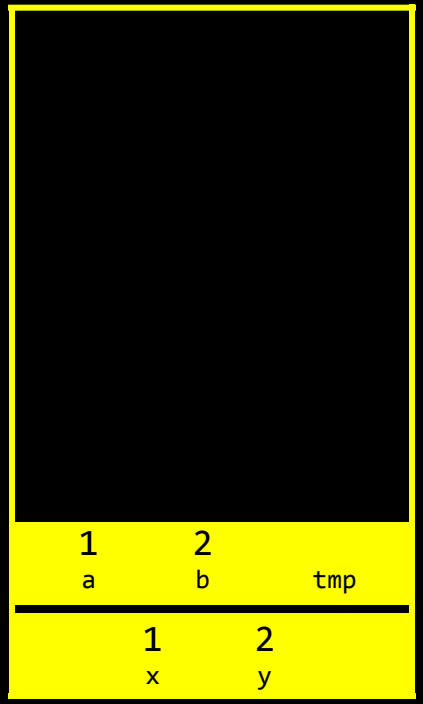

main


x
y

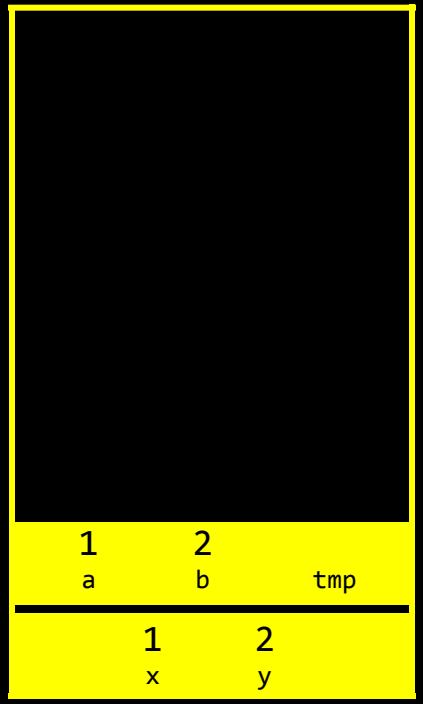

1
x
2
y


swap
1
x
2
y

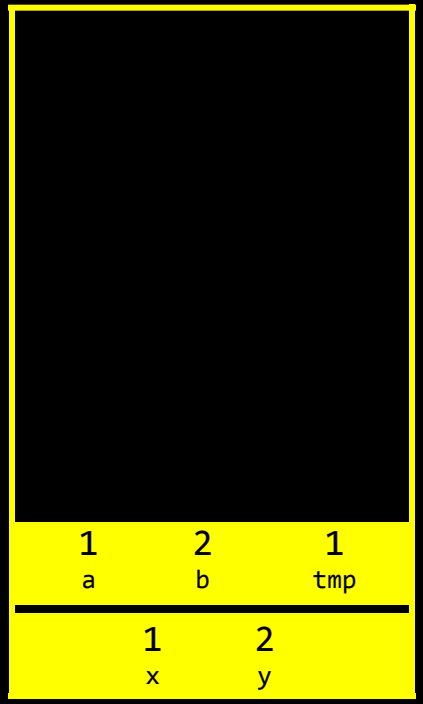

a
b
tmp
1
2
x
y


1
a
2
b
tmp
1
x
2
y

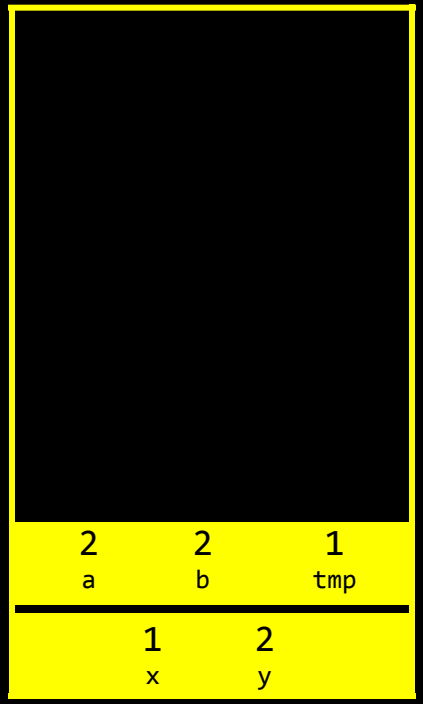
```
int tmp = a;
a = b;
b = tmp;
```



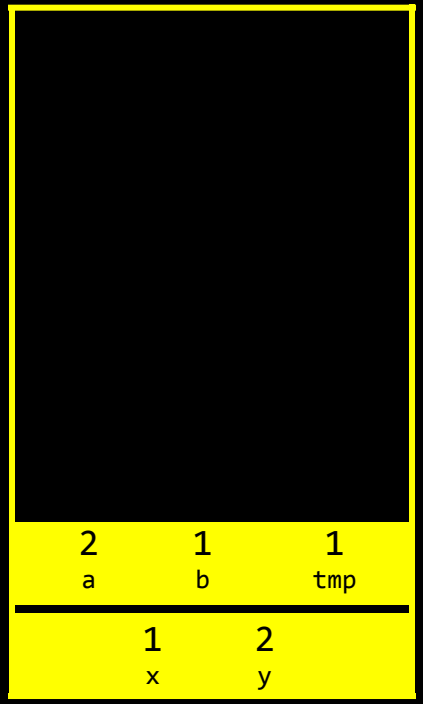
```
int tmp = a;
a = b;
b = tmp;
```

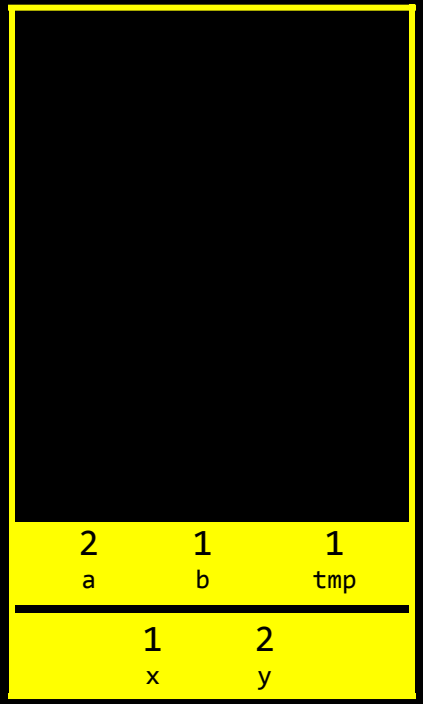


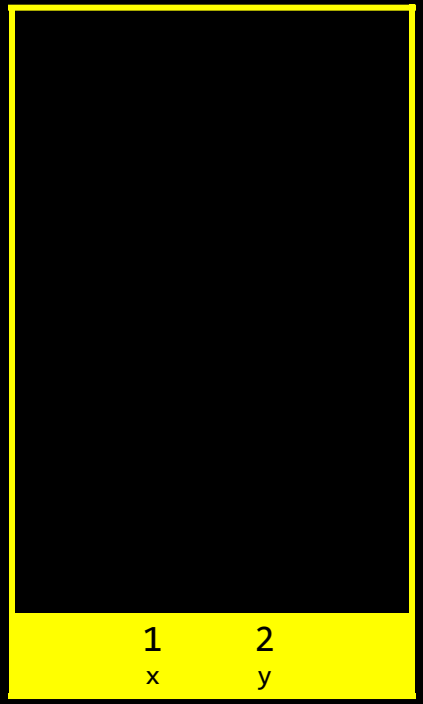
```
int tmp = a;
a = b;
b = tmp;
```



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

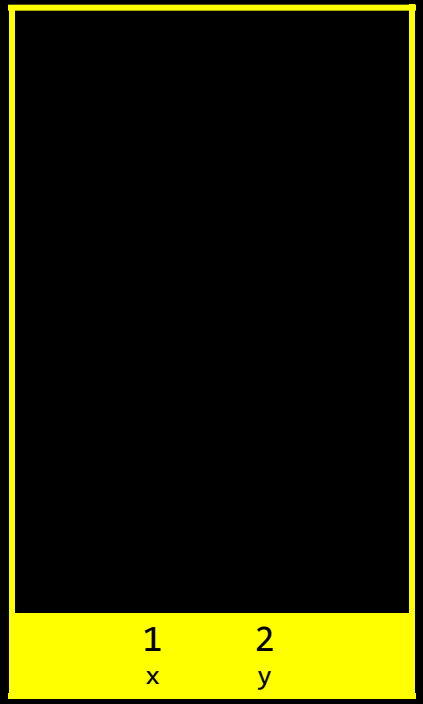


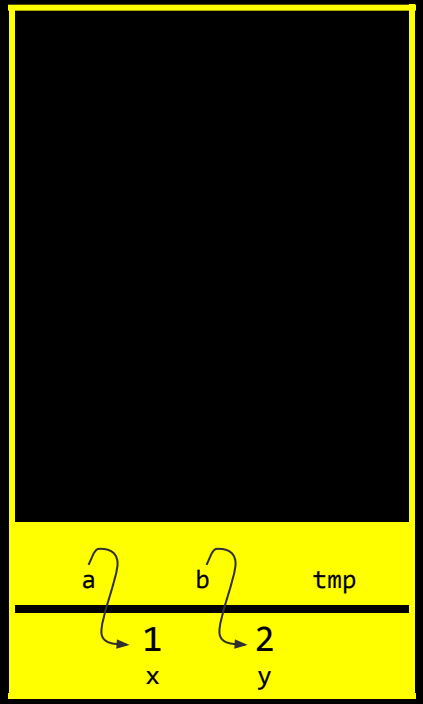

2
a
1
b
1
tmp
1
x
2
y


1
x
2
y

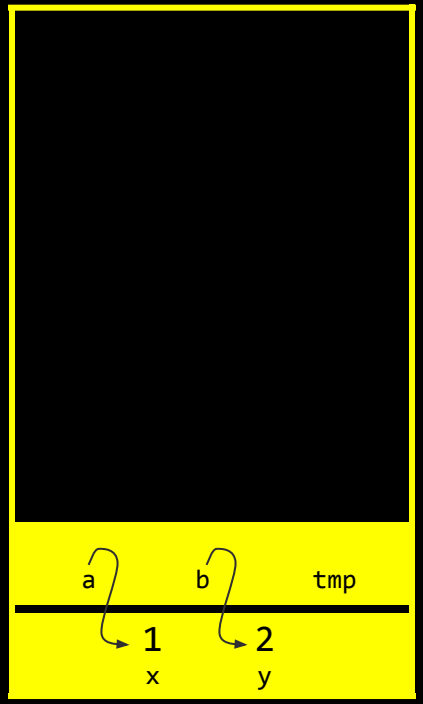
```
void swap(int a, int b)
{
    int tmp = a;
    a = b;
    b = tmp;
}
```

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

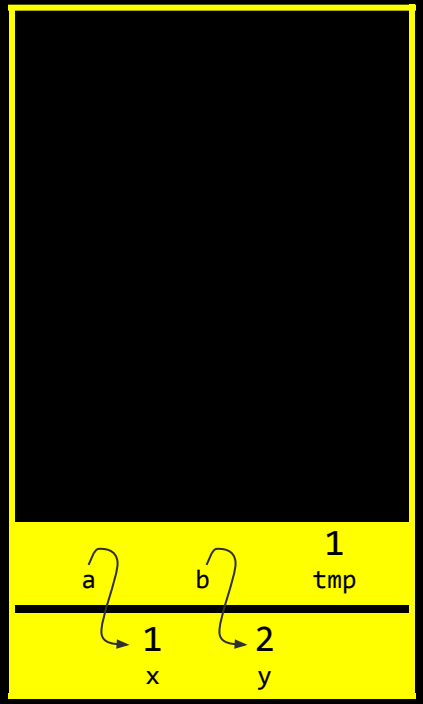

1
x
2
y


a
b
tmp
1
2
x
y

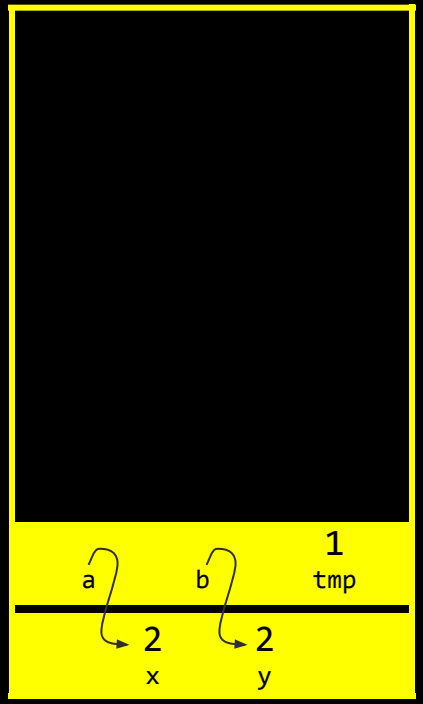
```
int tmp = *a;
*a = *b;
*b = tmp;
```



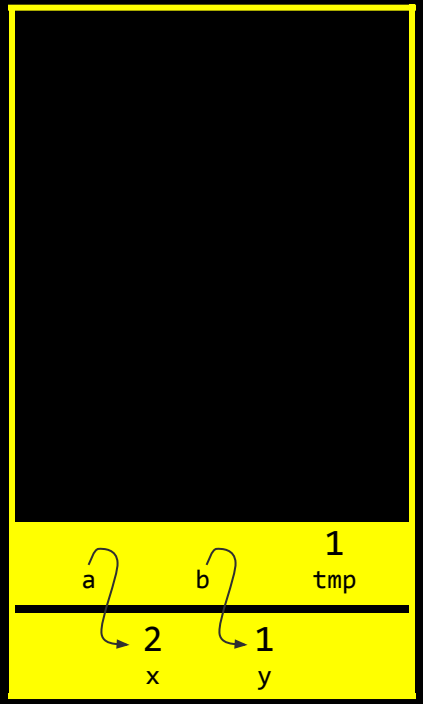
```
int tmp = *a;
*a = *b;
*b = tmp;
```

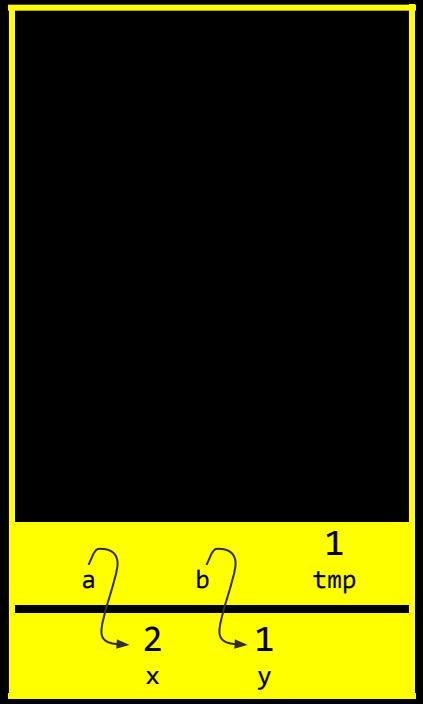


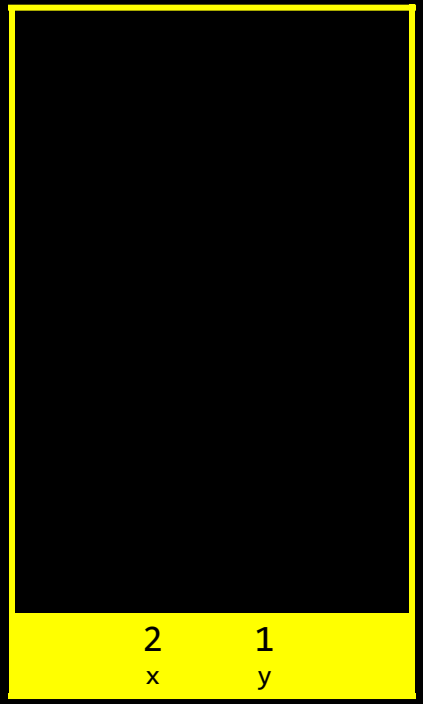
```
int tmp = *a;
*a = *b;
*b = tmp;
```



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

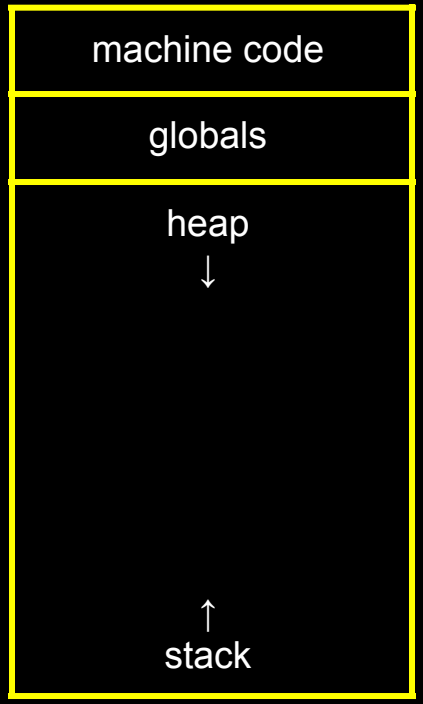


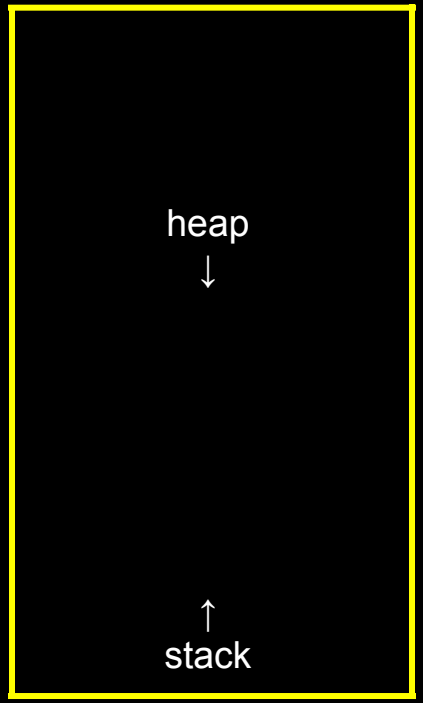

a
b
1
tmp
2
x
1
y


2
x
1
y

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

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


machine code
globals
heap
↓
↑
stack


heap
↓
↑
stack

heap overflow

stack overflow

buffer overflow

```
get_char

get_double

get_float

get_int

get_long

get_string

...
```

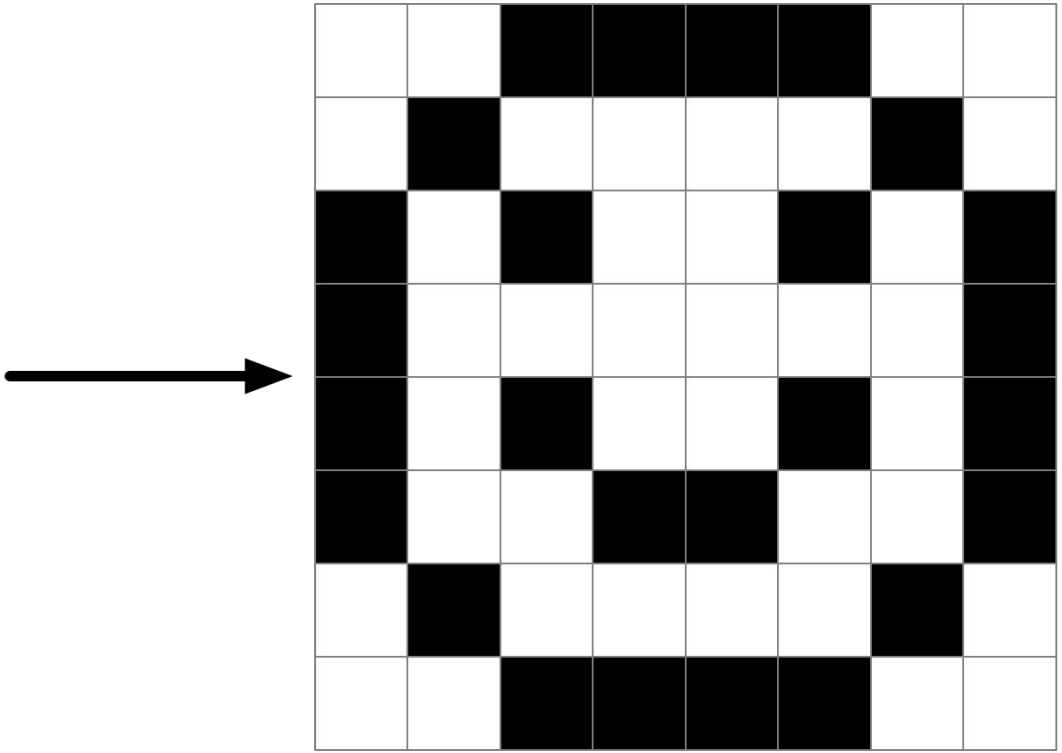
scanf

...

file I/O

```
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
```

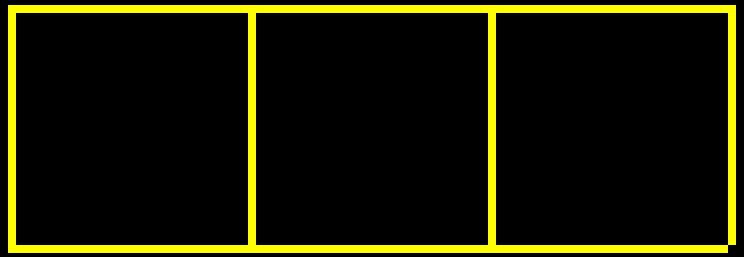
```
1 1 0 0 0 0 1 1
1 0 1 1 1 1 0 1
0 1 0 1 1 0 1 0
0 1 1 1 1 1 1 0
0 1 0 1 1 0 1 0
0 1 1 0 0 1 1 0
1 0 1 1 1 1 0 1
1 1 0 0 0 0 1 1
```

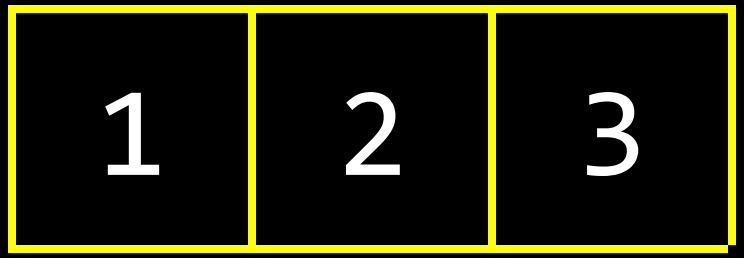


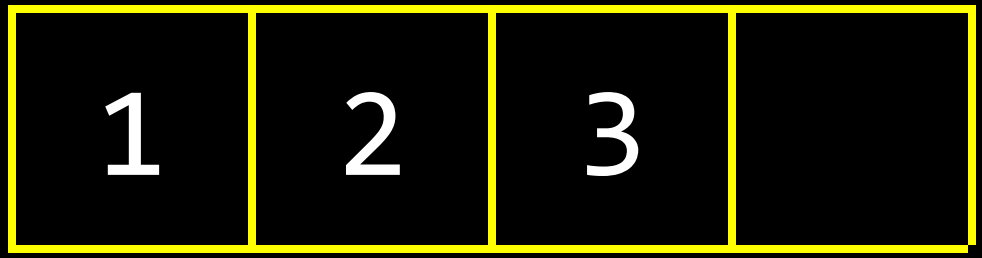
JPEG

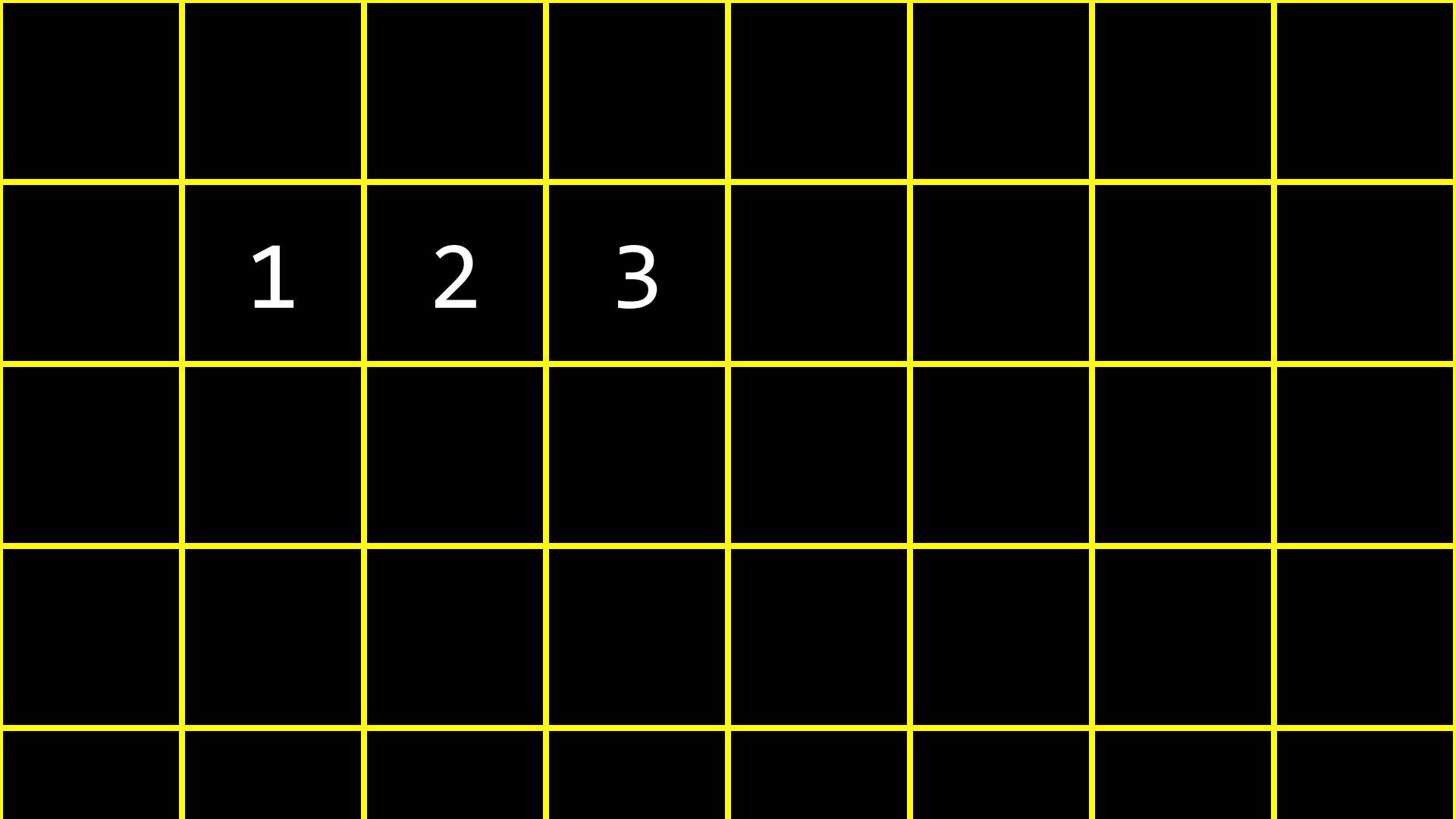
BMP

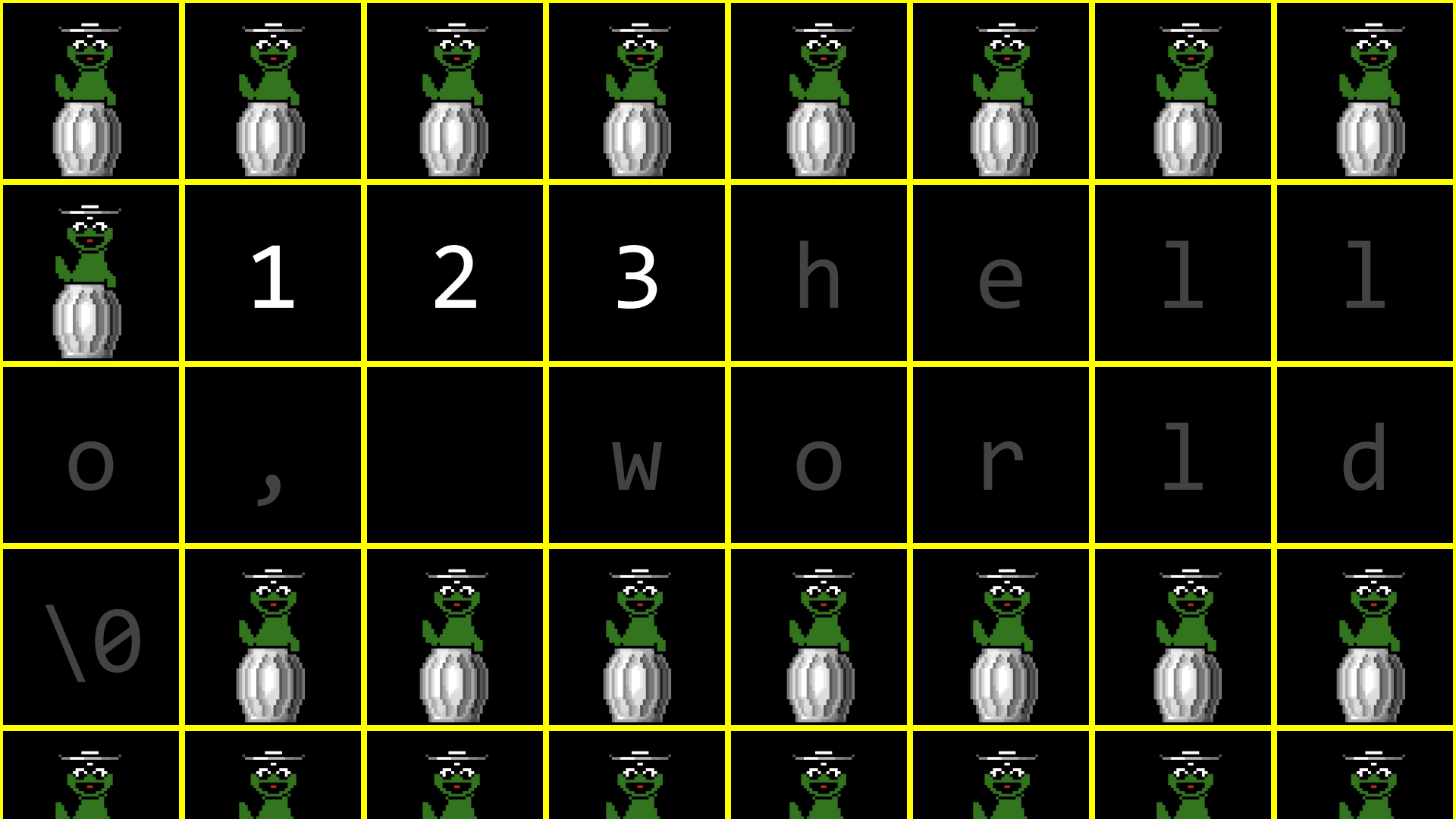
arrays




1
2
3


1
2
3


1
2
3


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

1	2	3

1	2	3

1			

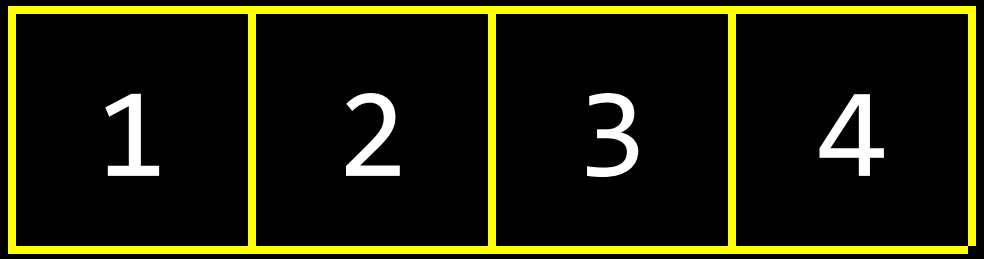
1	2	3

1	2		

1	2	3

1	2	3	

1	2	3	


1
2
3
4

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$ search

$O(1)$

$O(n^2)$

$O(n \log n)$

$O(n)$ insert

$O(\log n)$ search

$O(1)$

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

$\Omega(\log n)$

$\Omega(1)$

data structures

struct

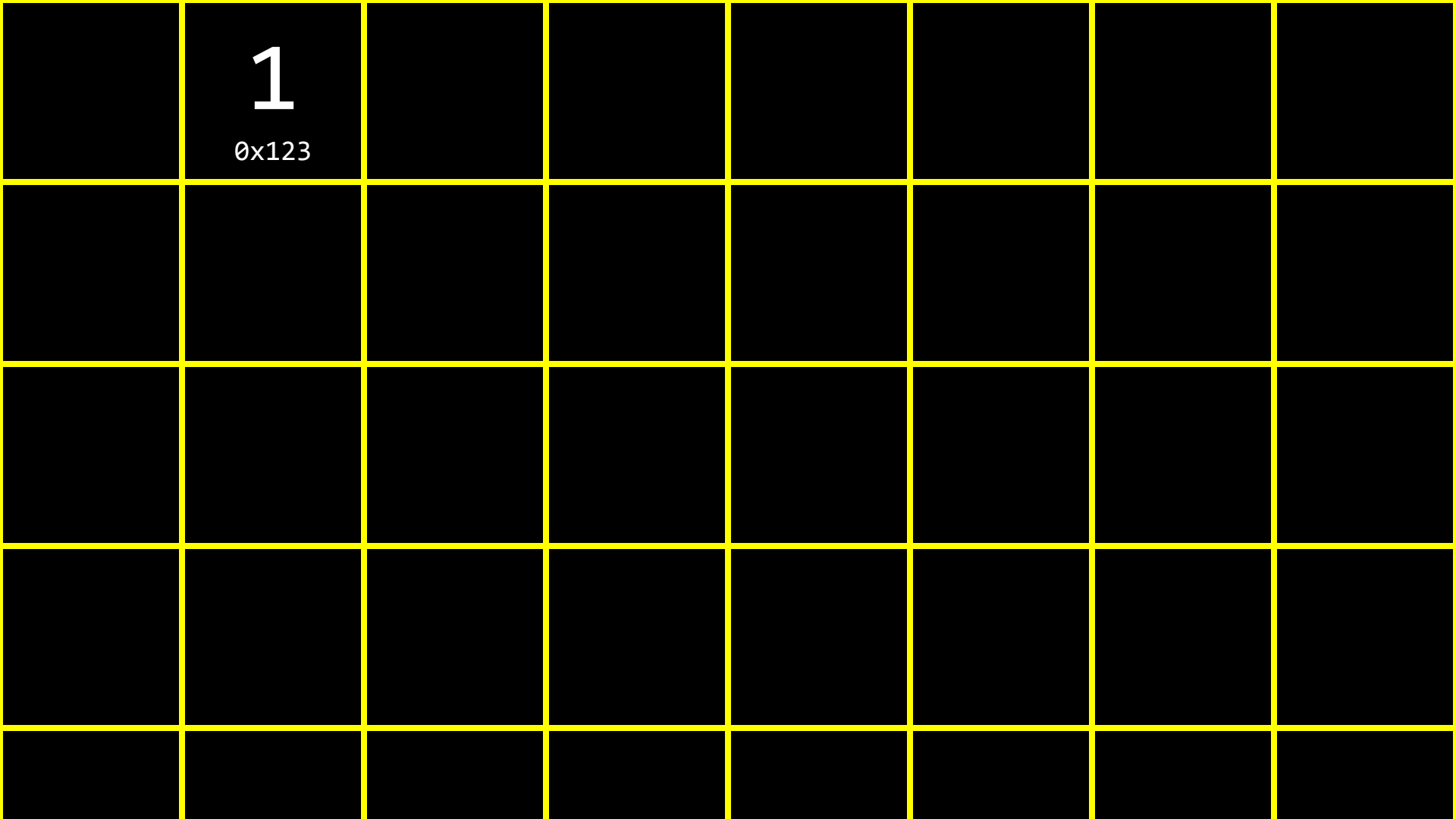
.

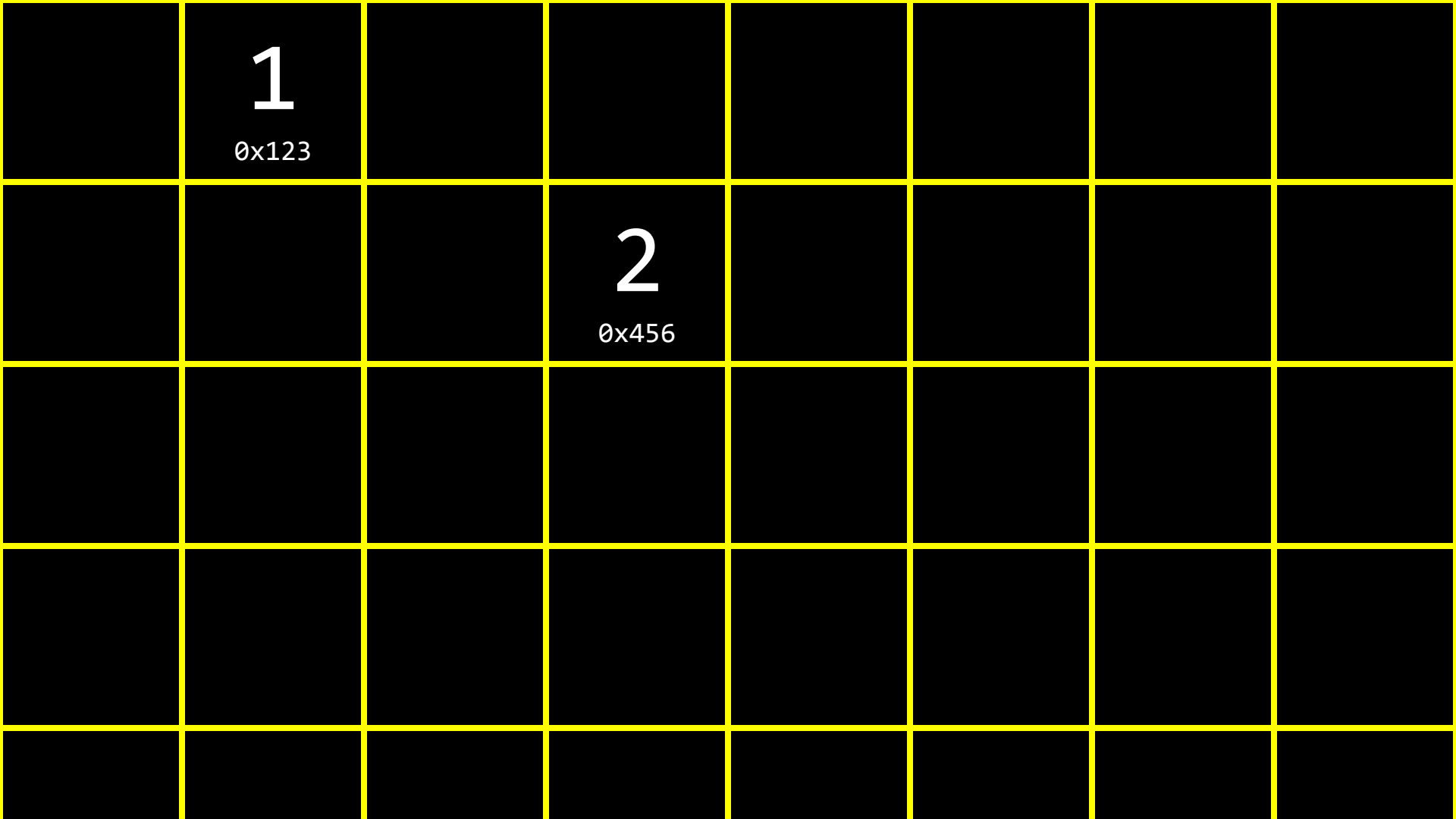
*

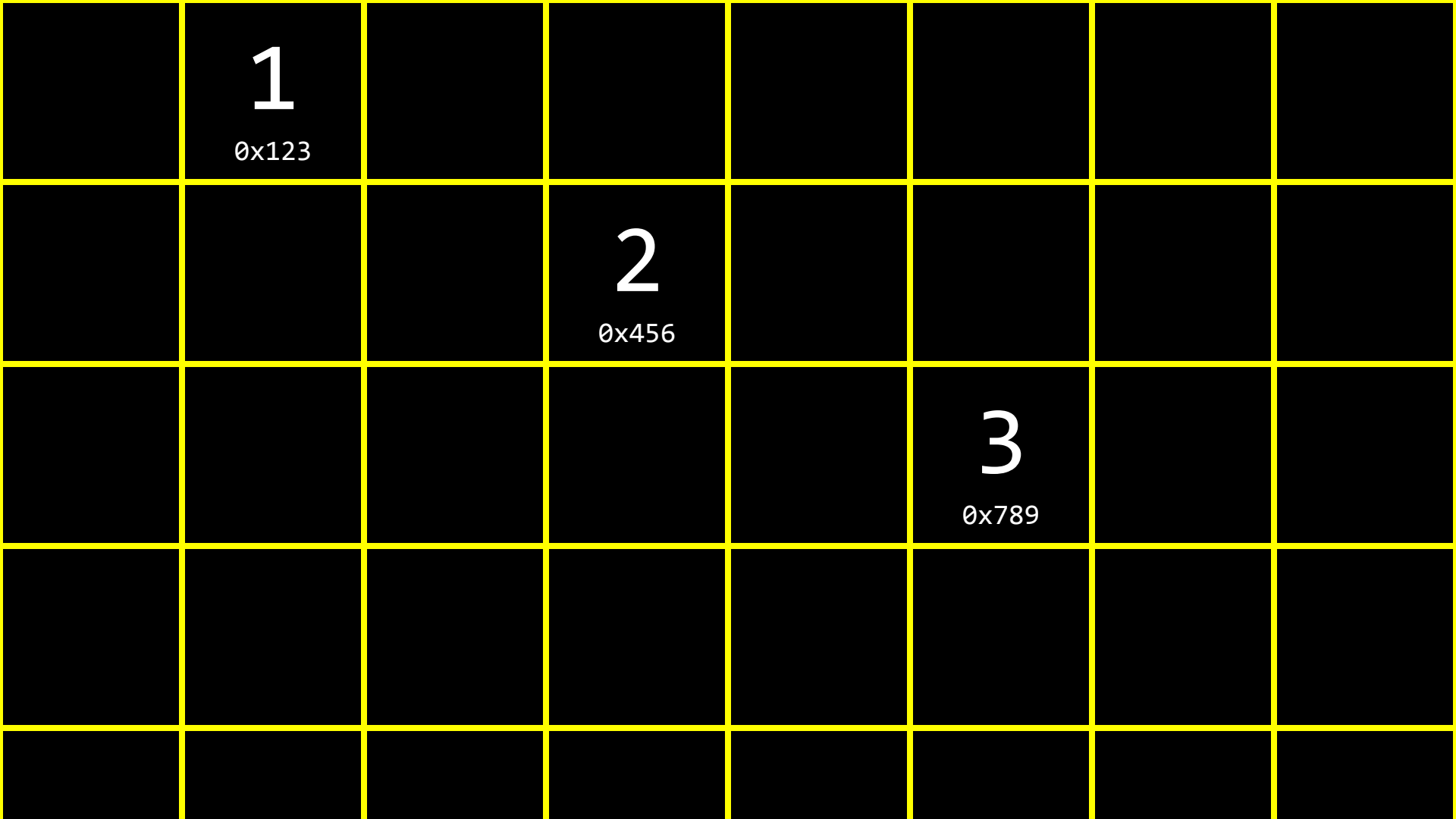
struct

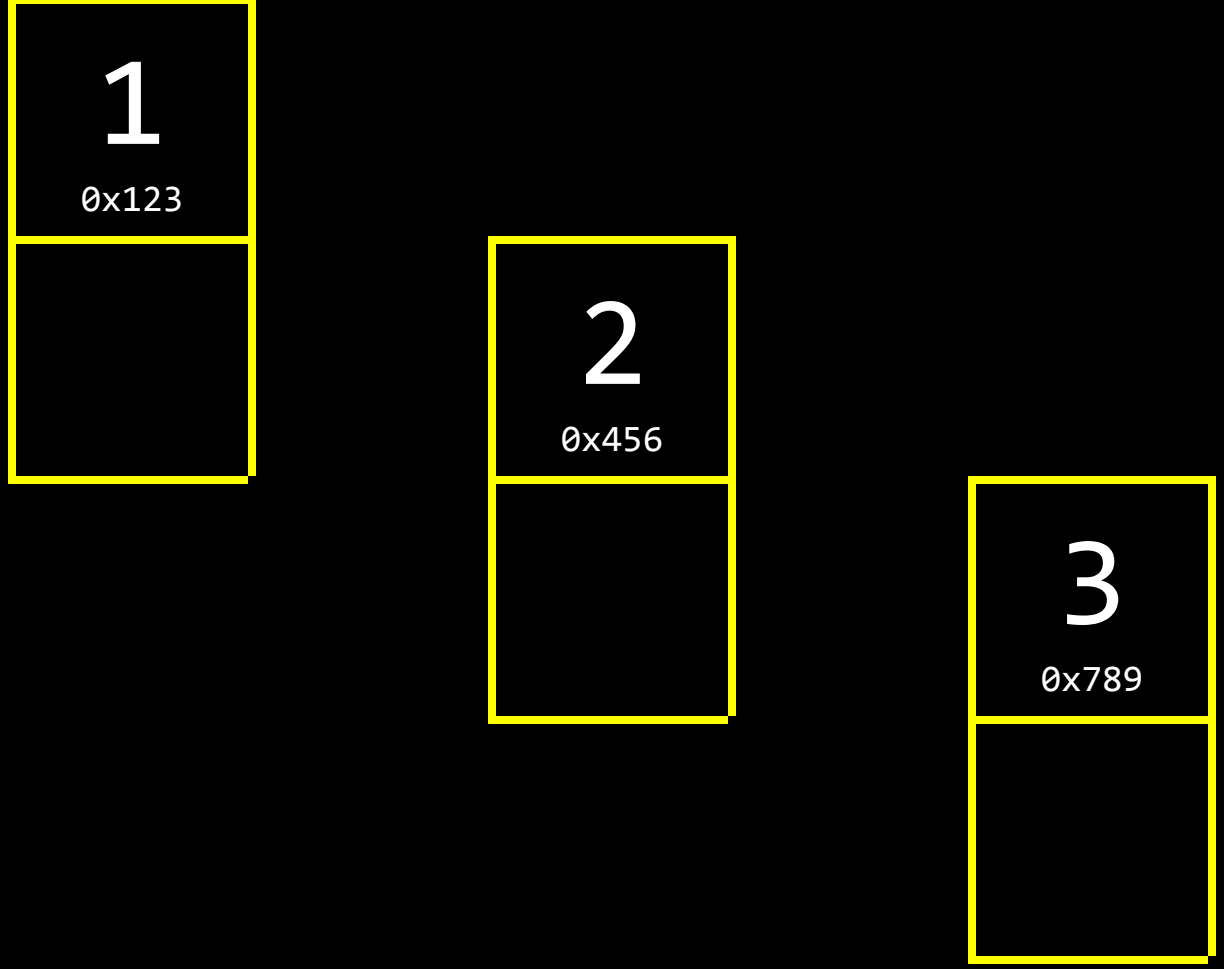
->

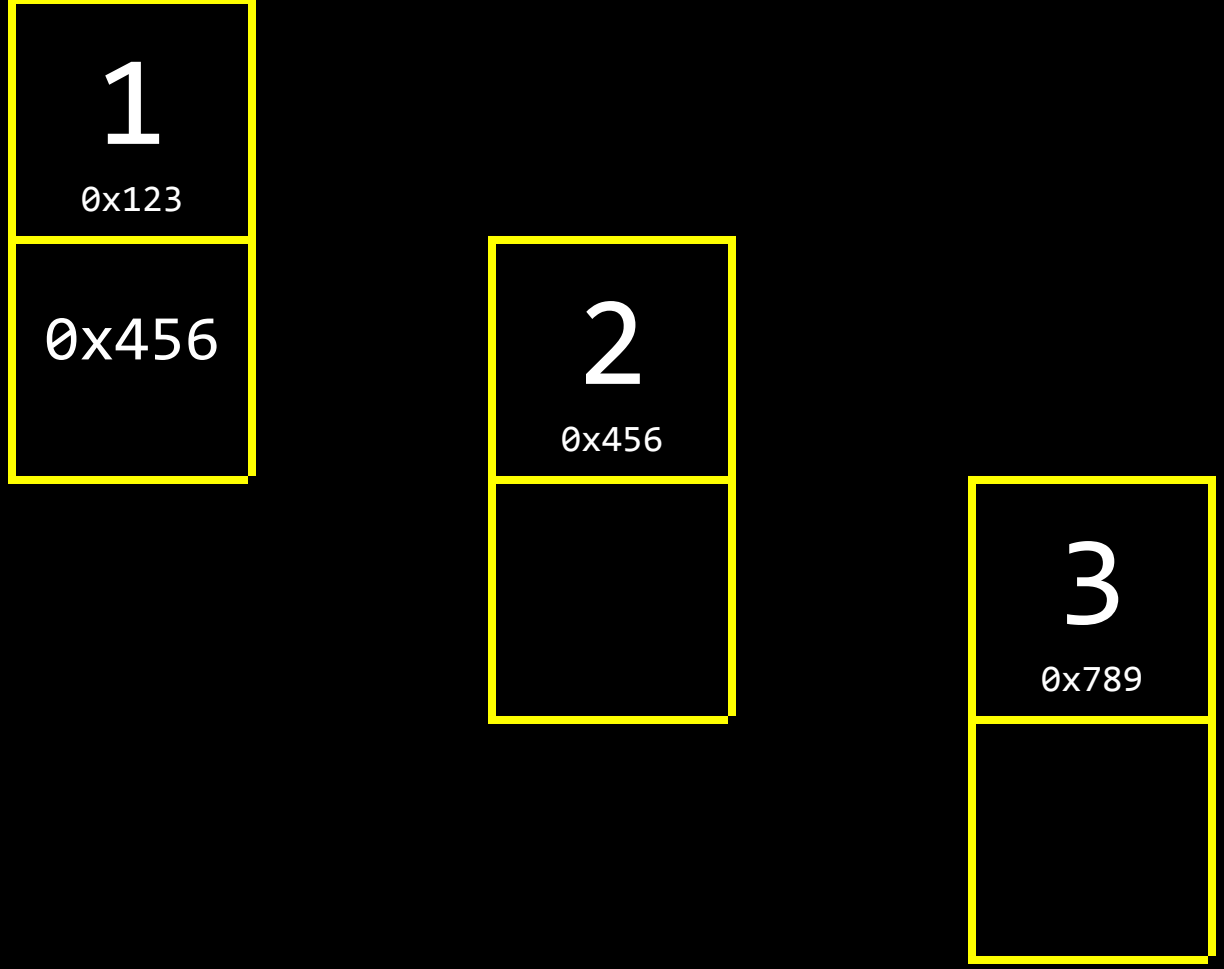
linked lists

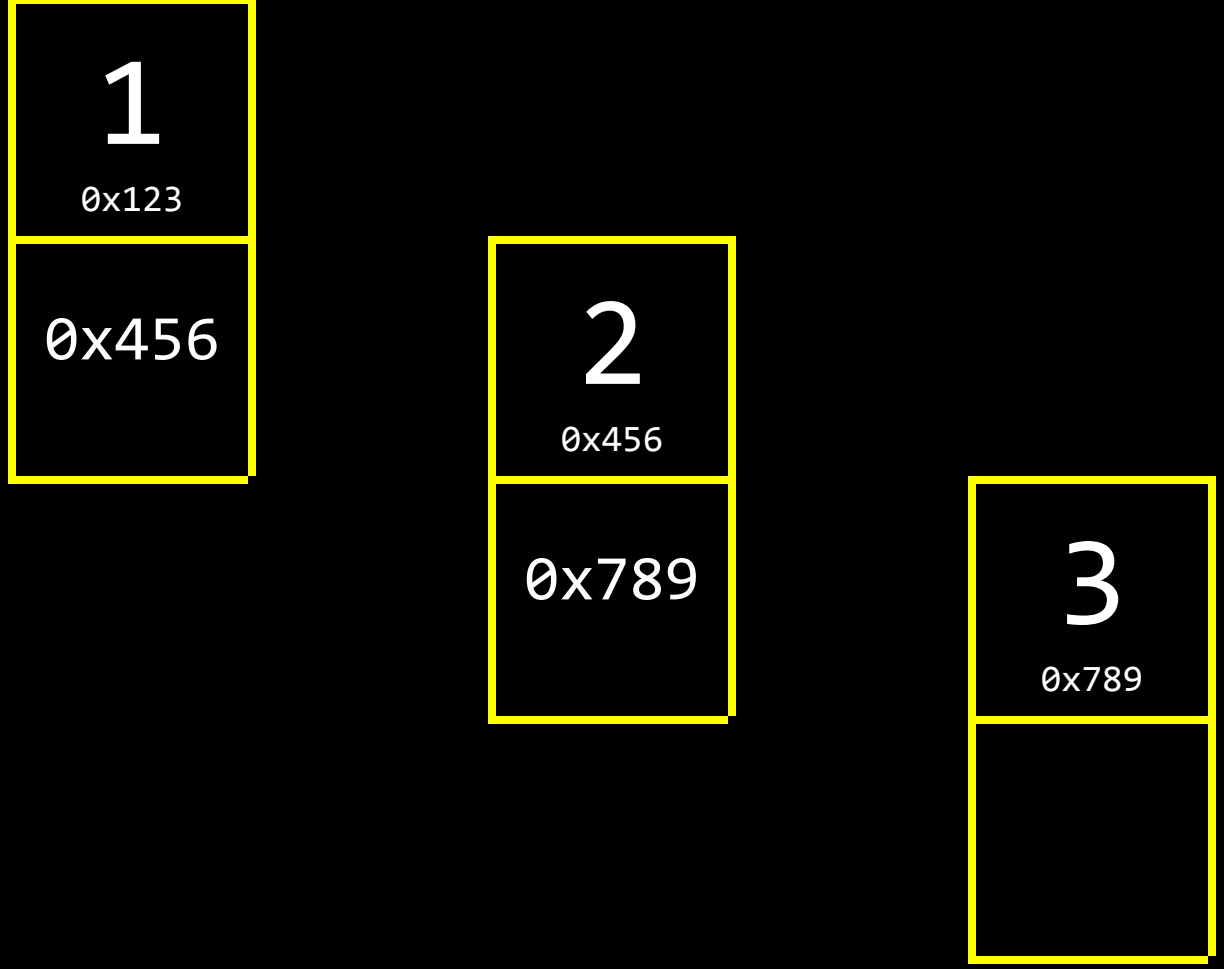

1
0x123

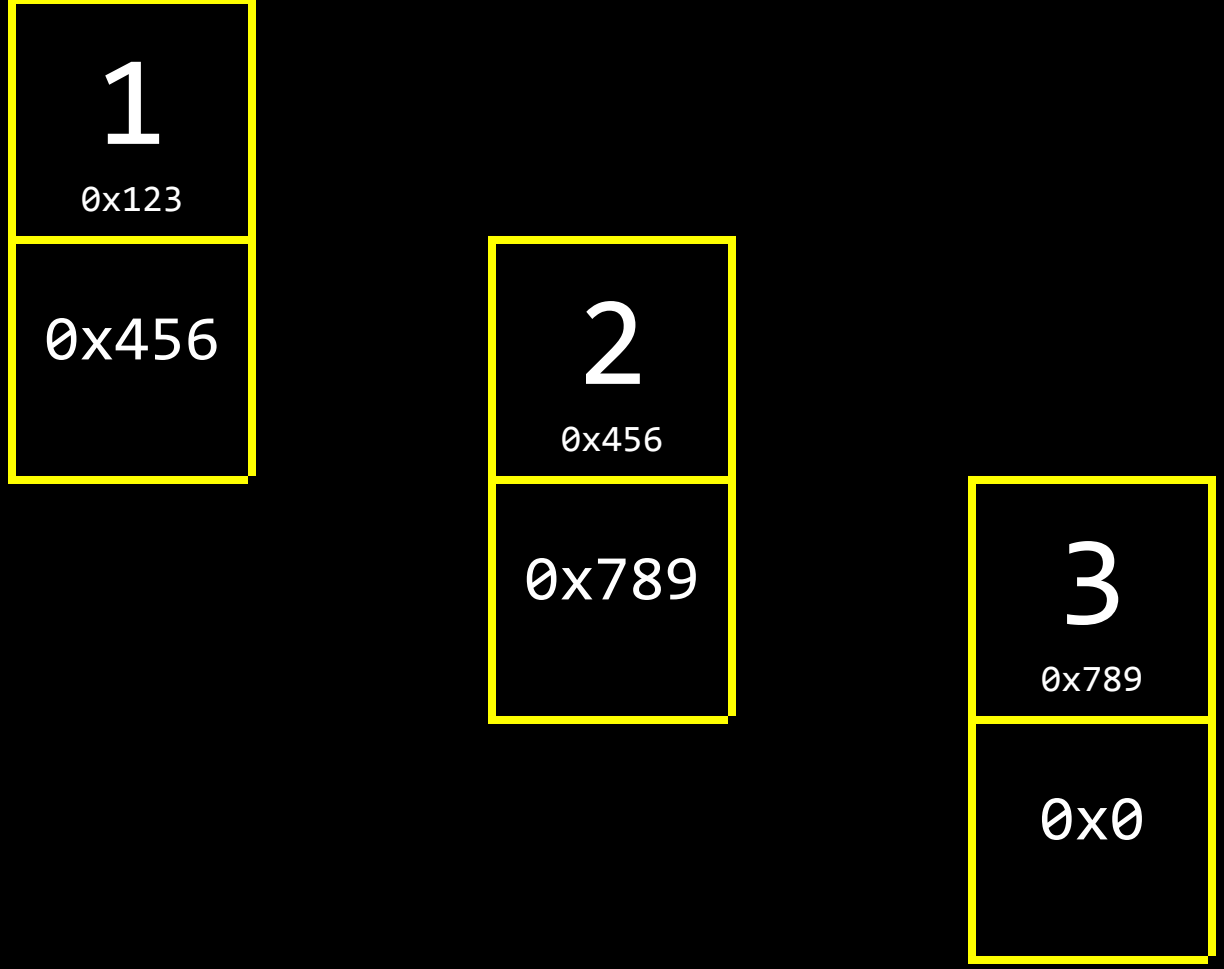

1
0x123
2
0x456

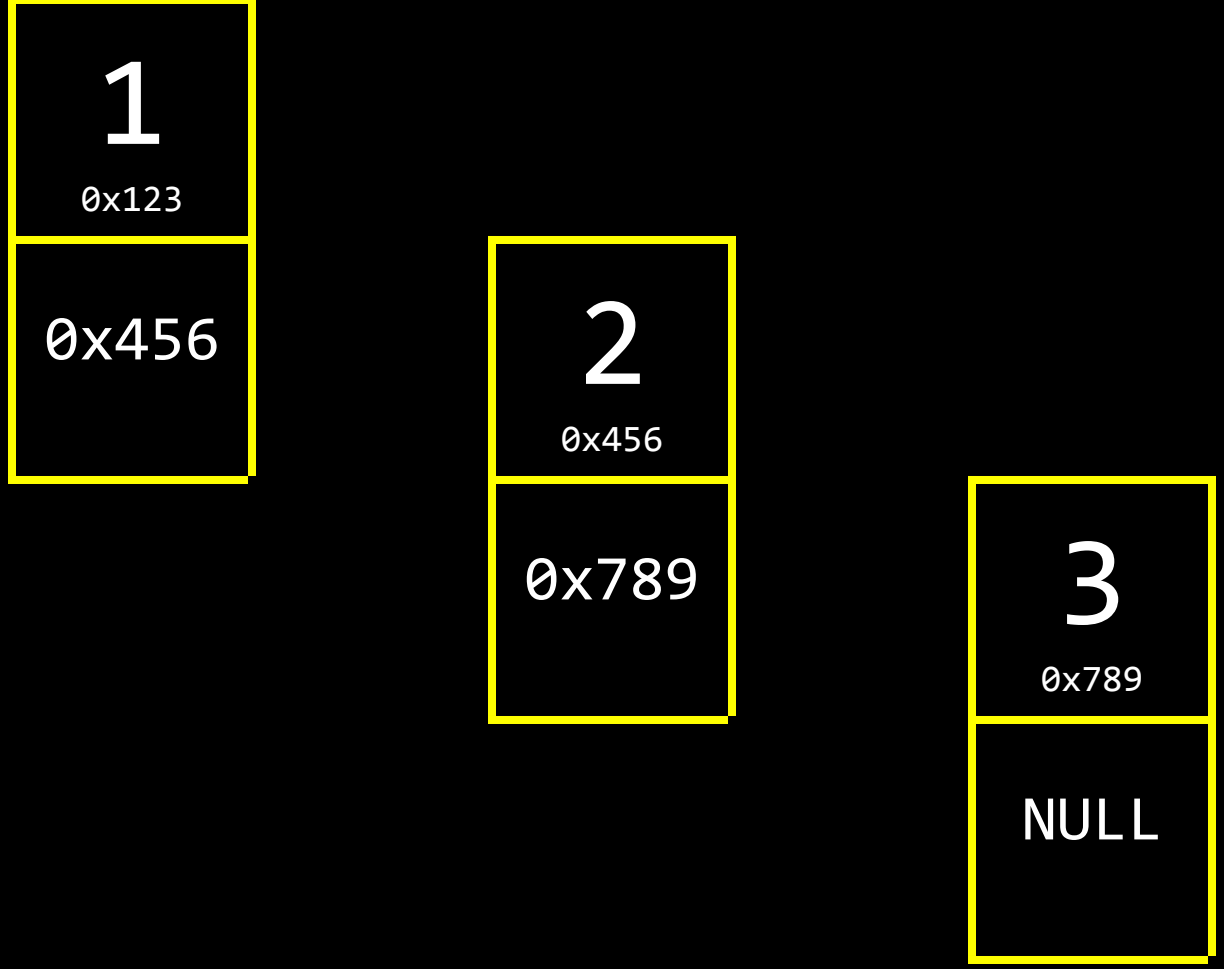

1
0x123
2
0x456
3
0x789

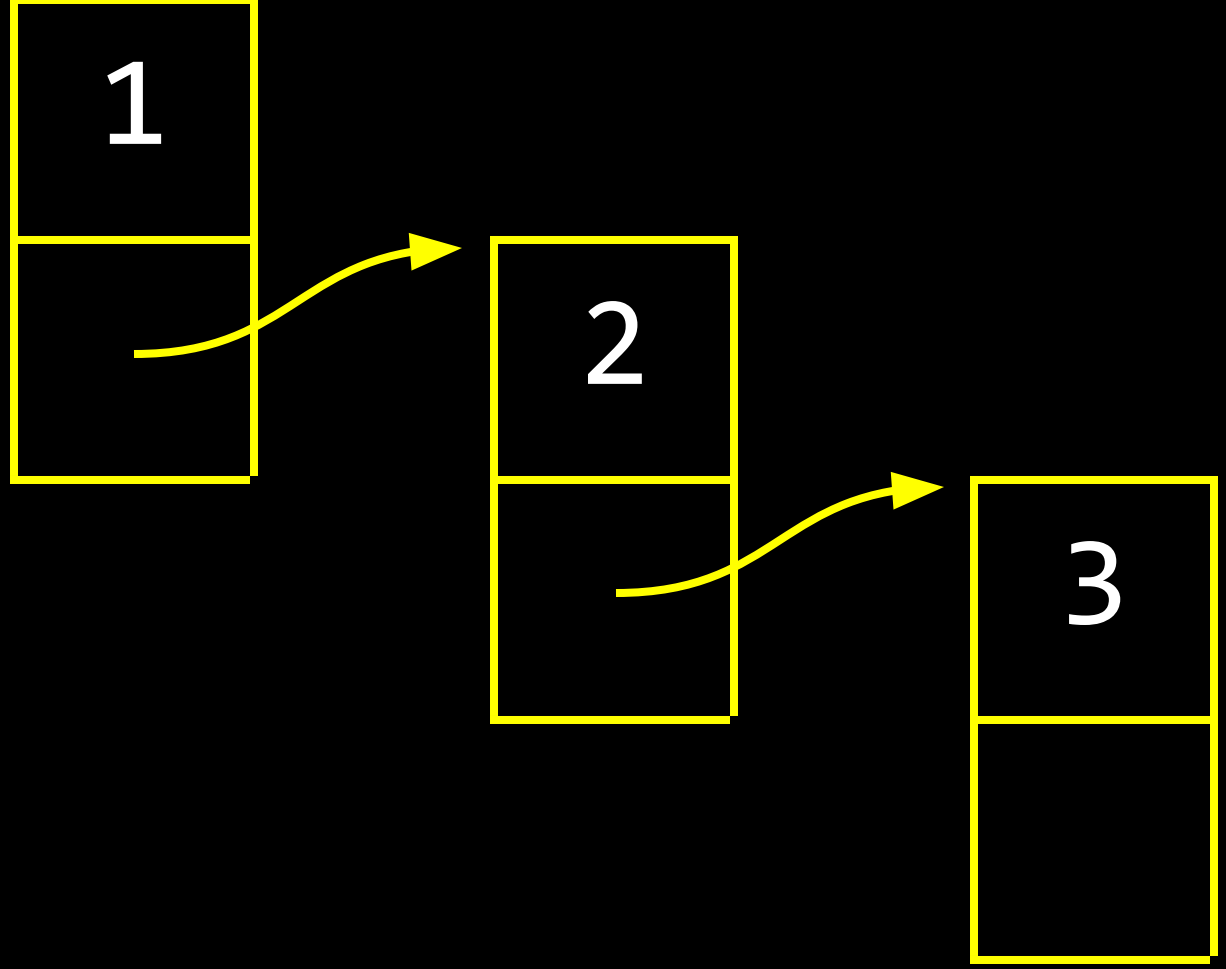

1
0x123
2
0x456
3
0x789


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


1
2
3

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

```
typedef struct
{


}
person;
```

```
typedef struct
{


}
node;
```

```
typedef struct
{
    int number;

}
node;
```

```
typedef struct
{
    int number;
    node *next;
}
node;
```

```
typedef struct node
{
    int number;
    node *next;
}
node;
```

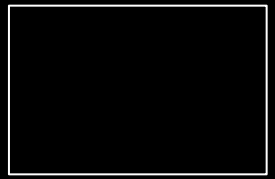
```
typedef struct node
{
    int number;
    struct node *next;
}
node;
```

```
node *list;
```

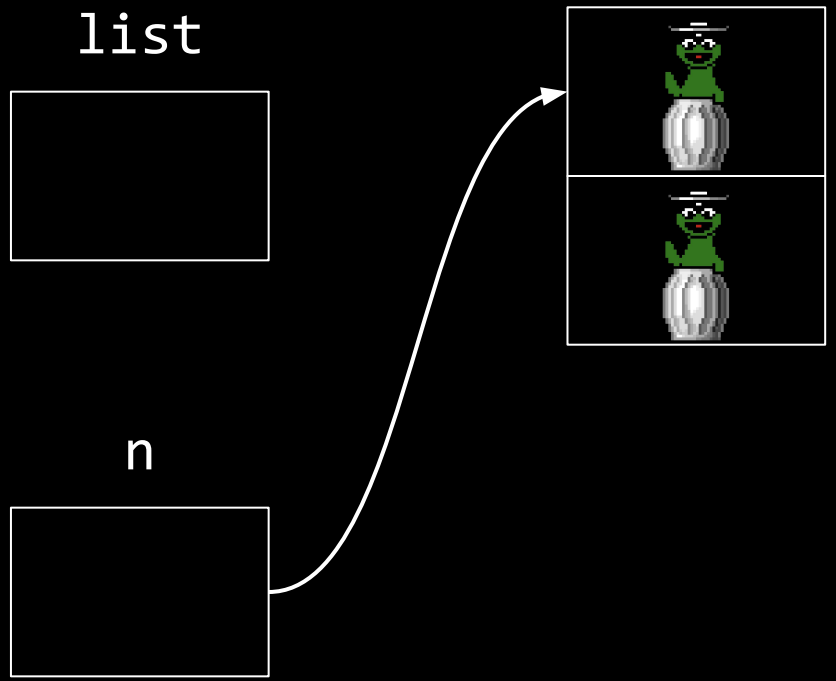
list

```
node *list = NULL;
```

list

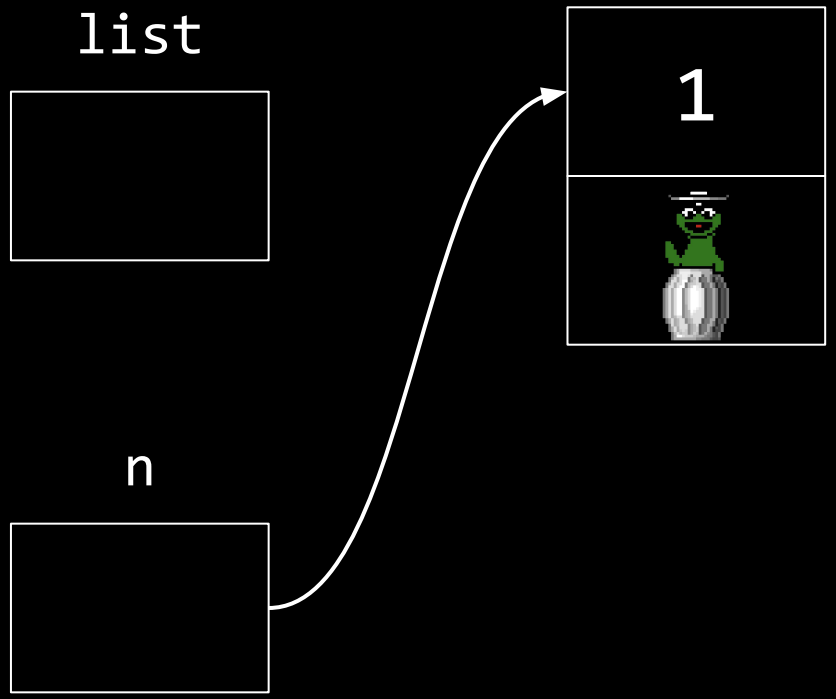


```
node *n = malloc(sizeof(node));
```

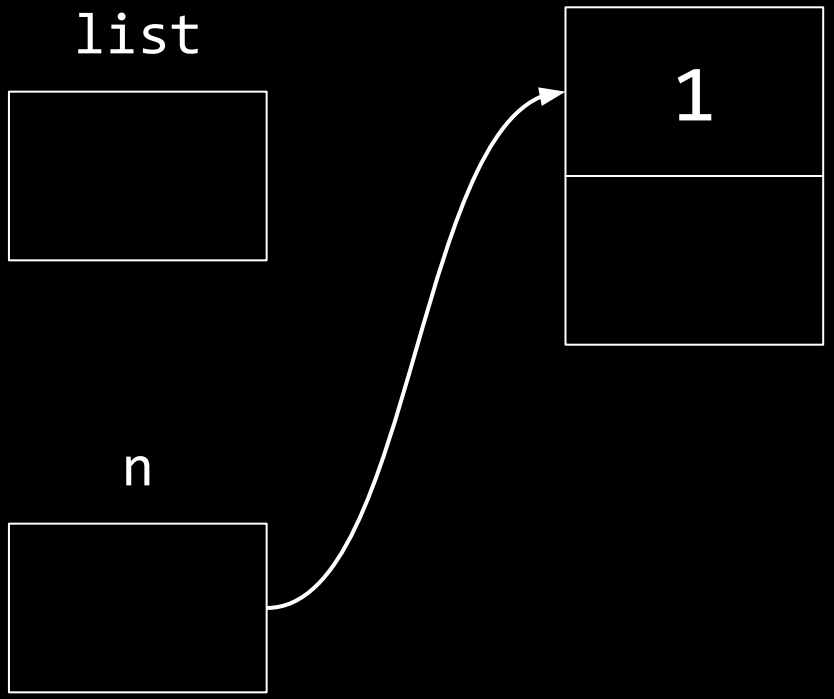

list
n

```
if (n != NULL)
{
    (*n).number = 1;
}
```

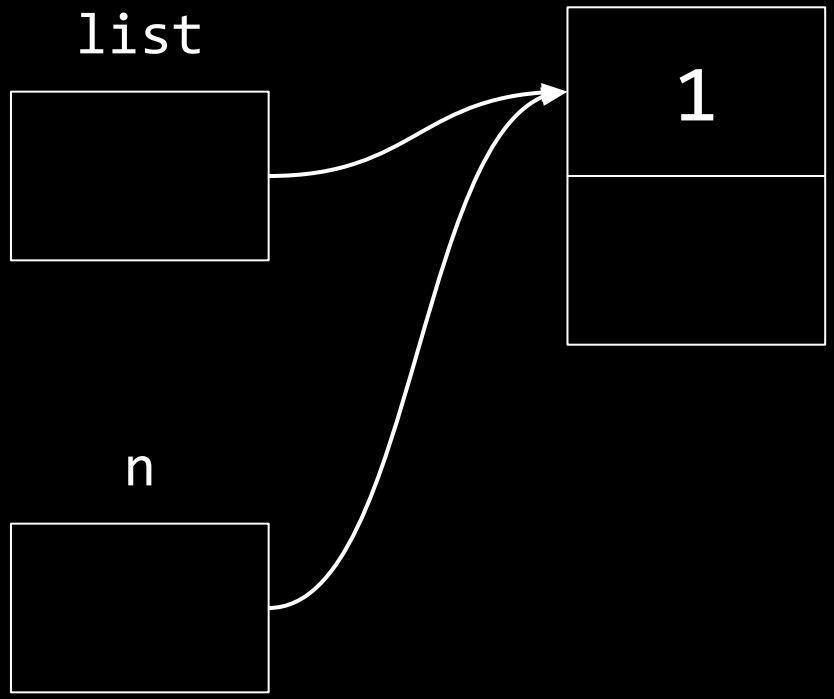
```
if (n != NULL)
{
    n->number = 1;
}
```

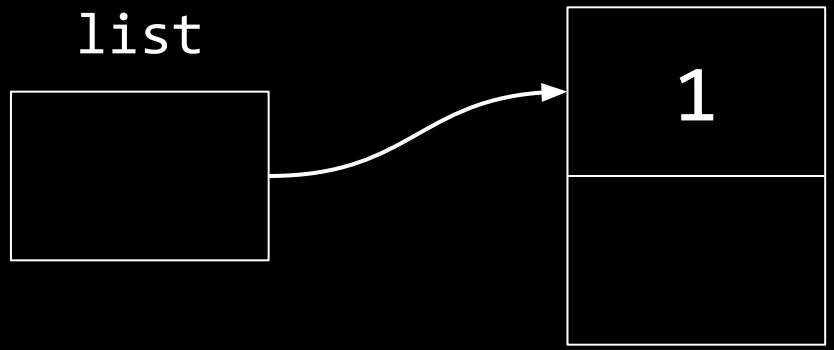

list
1
n

```
if (n != NULL)
{
    n->next = NULL;
}
```

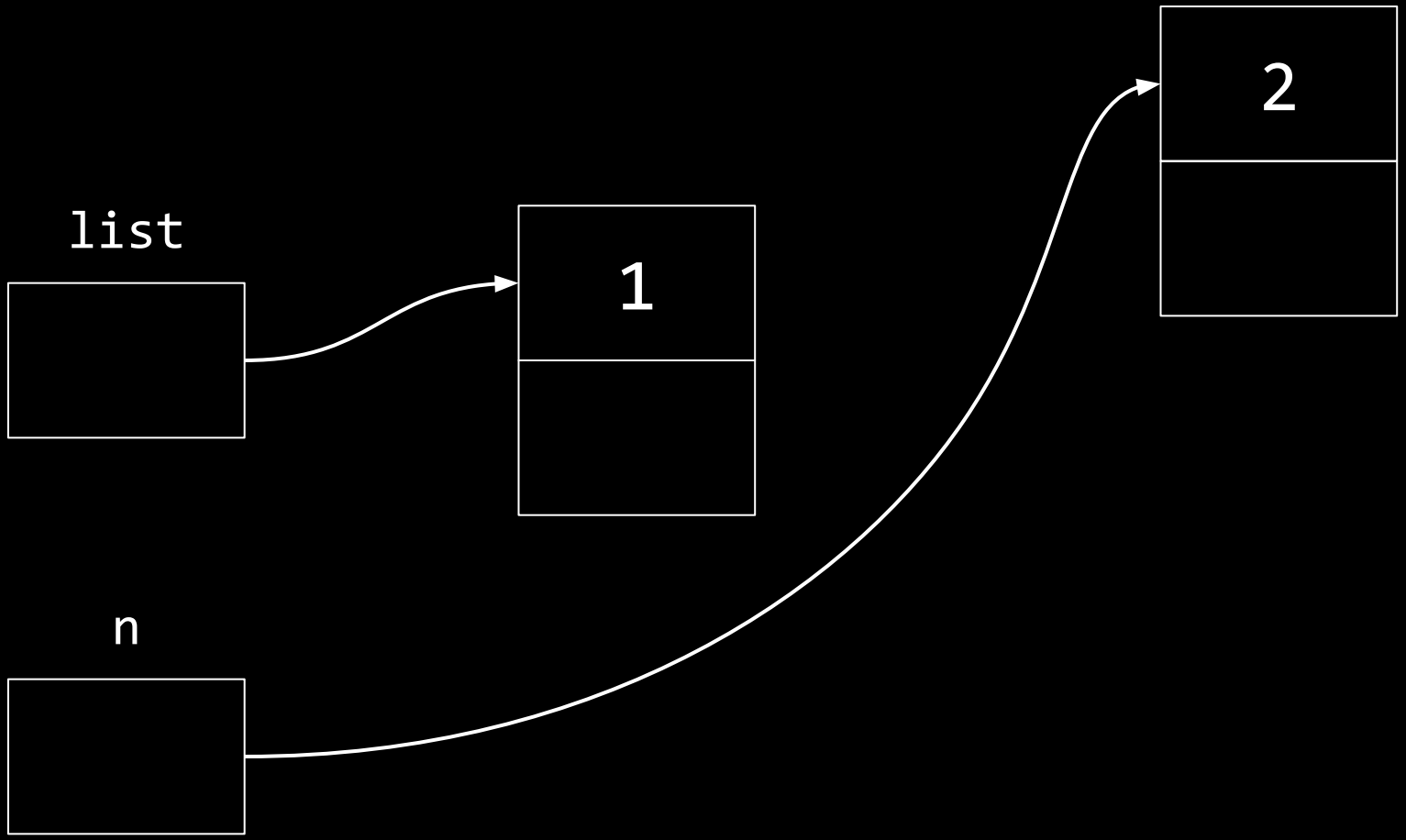

list
1
n

```
list = n;
```

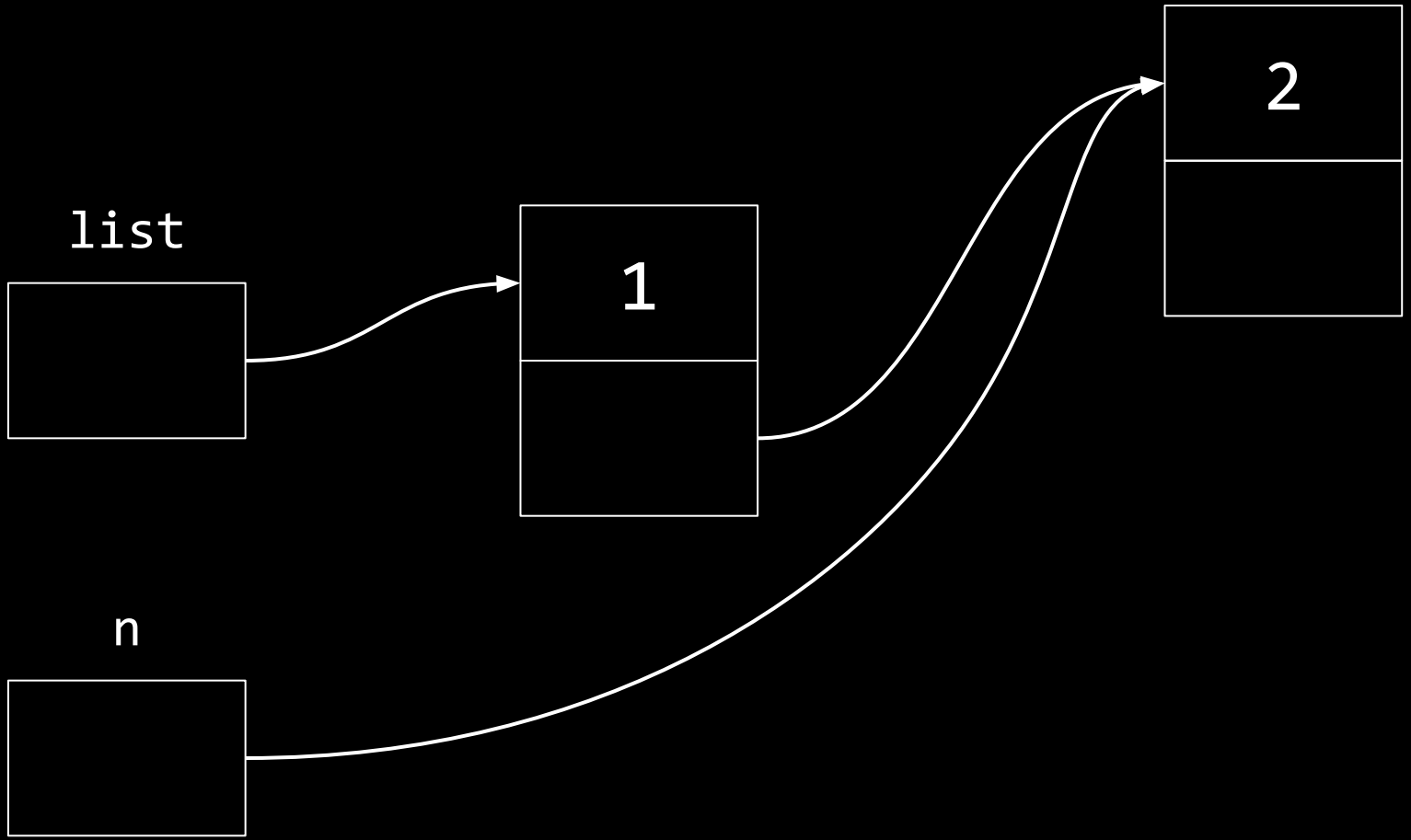

list
1
n

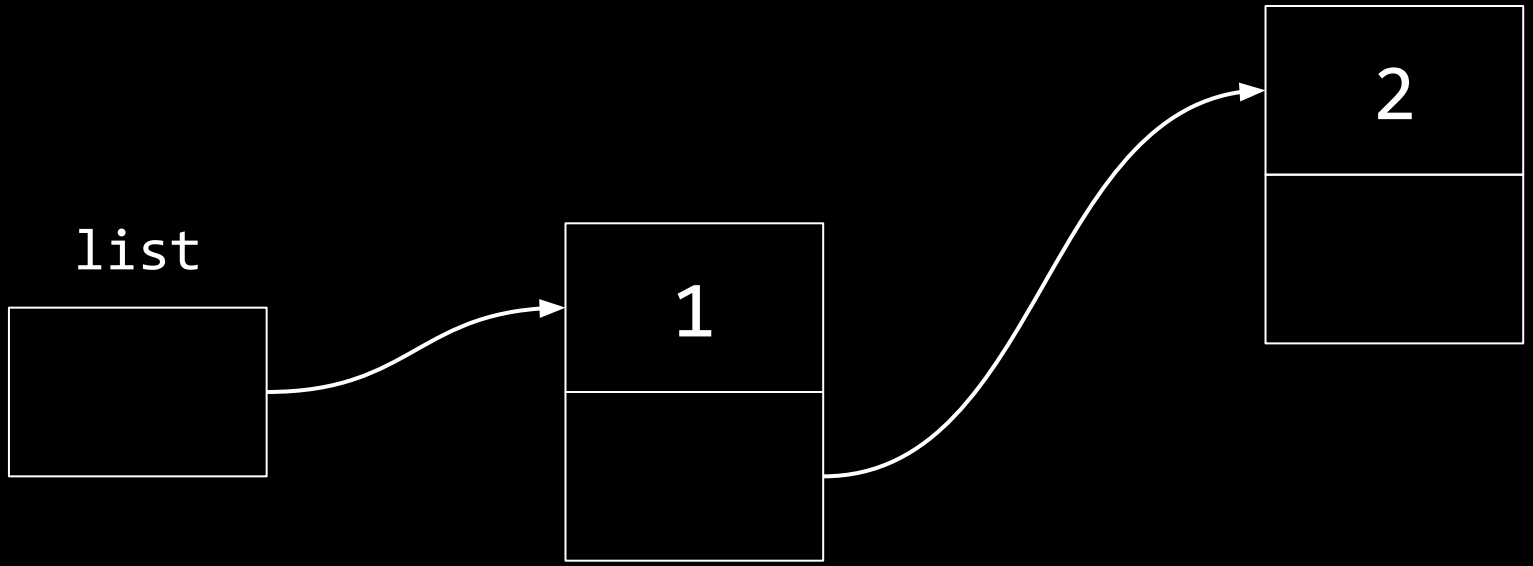

list
1

```
node *n = malloc(sizeof(node));
if (n != NULL)
{
    n->number = 2;
    n->next = NULL;
}
```

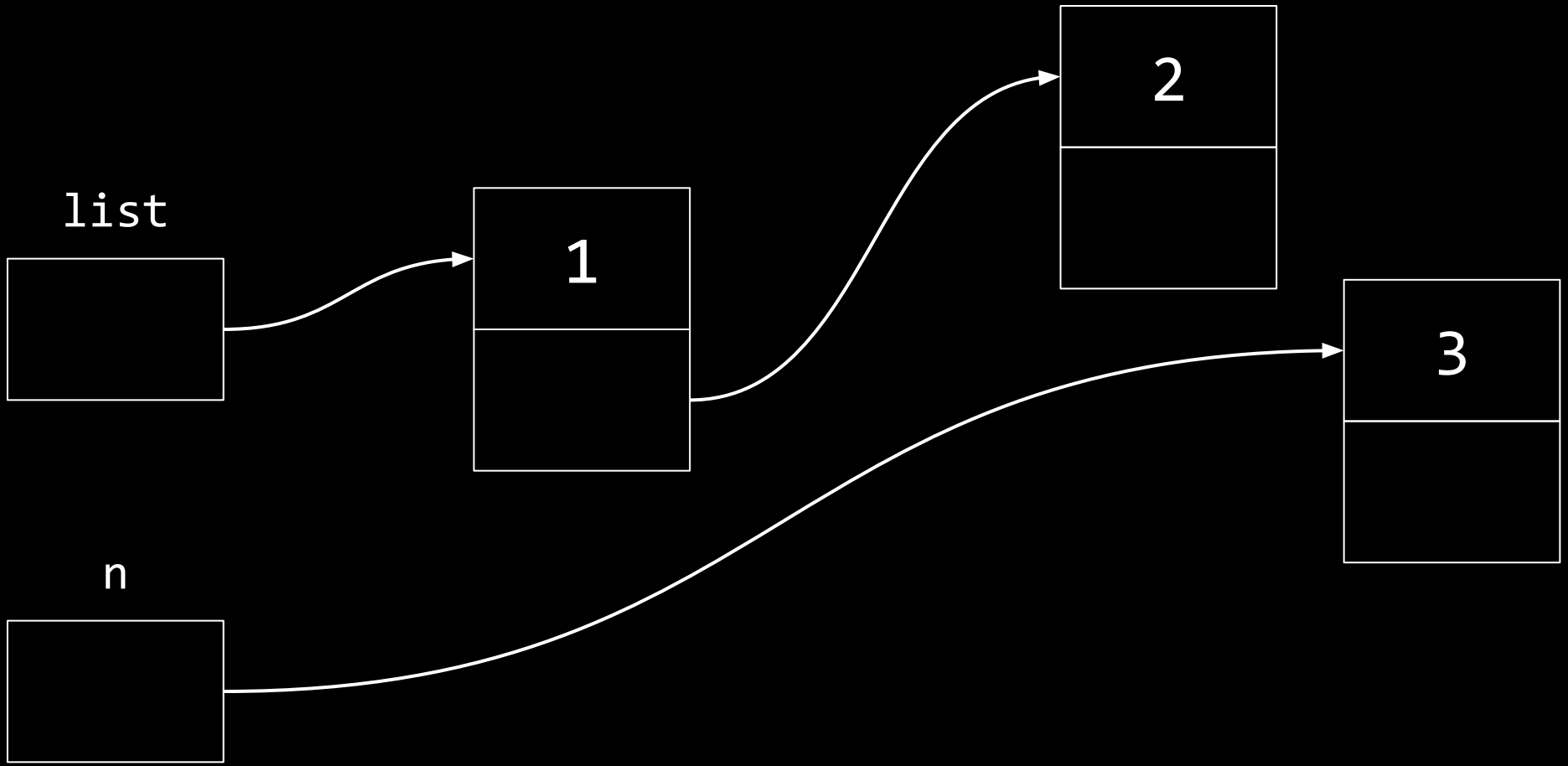

list
1
2
n

```
list->next = n;
```

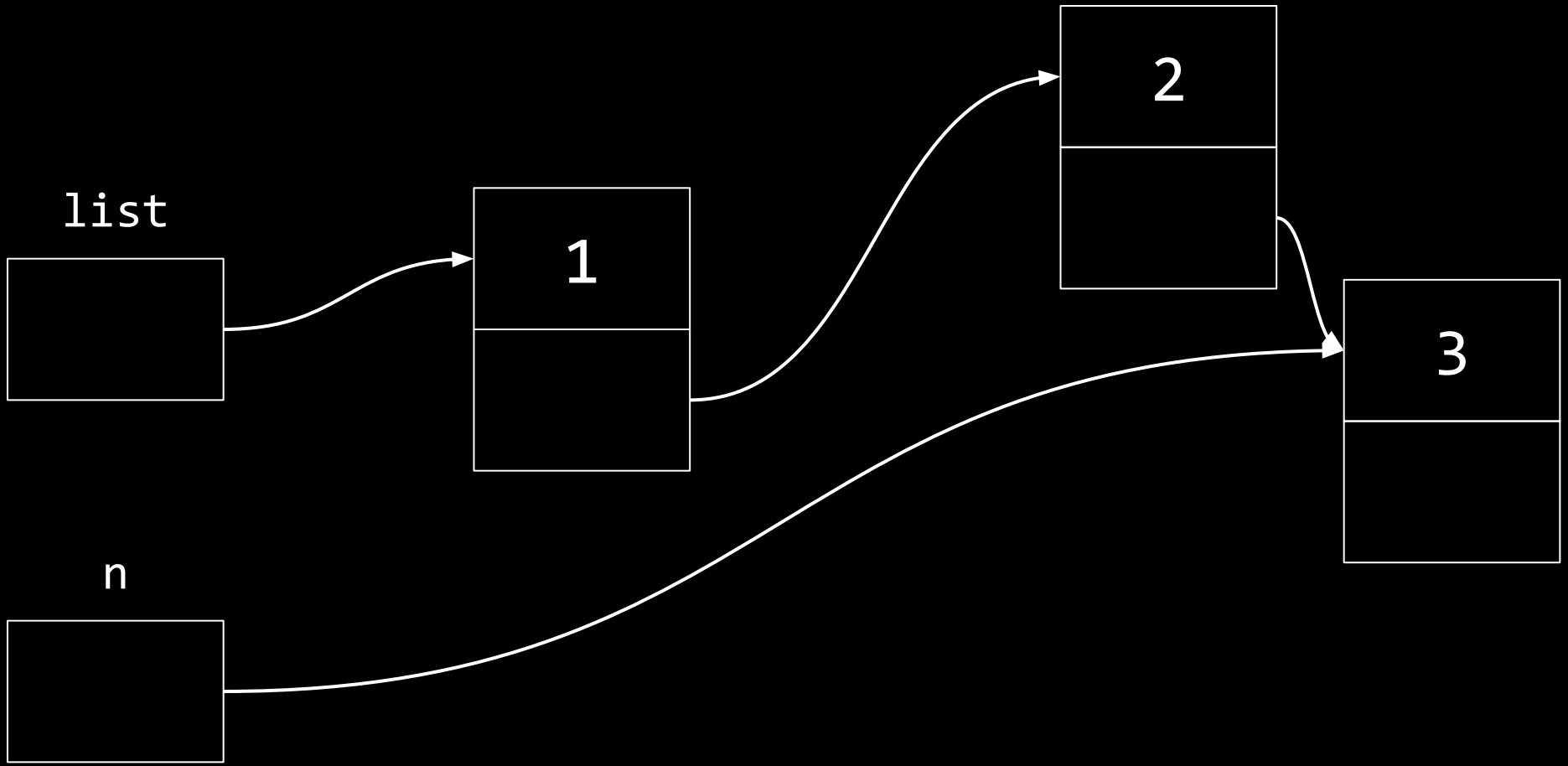

list
1
2
n

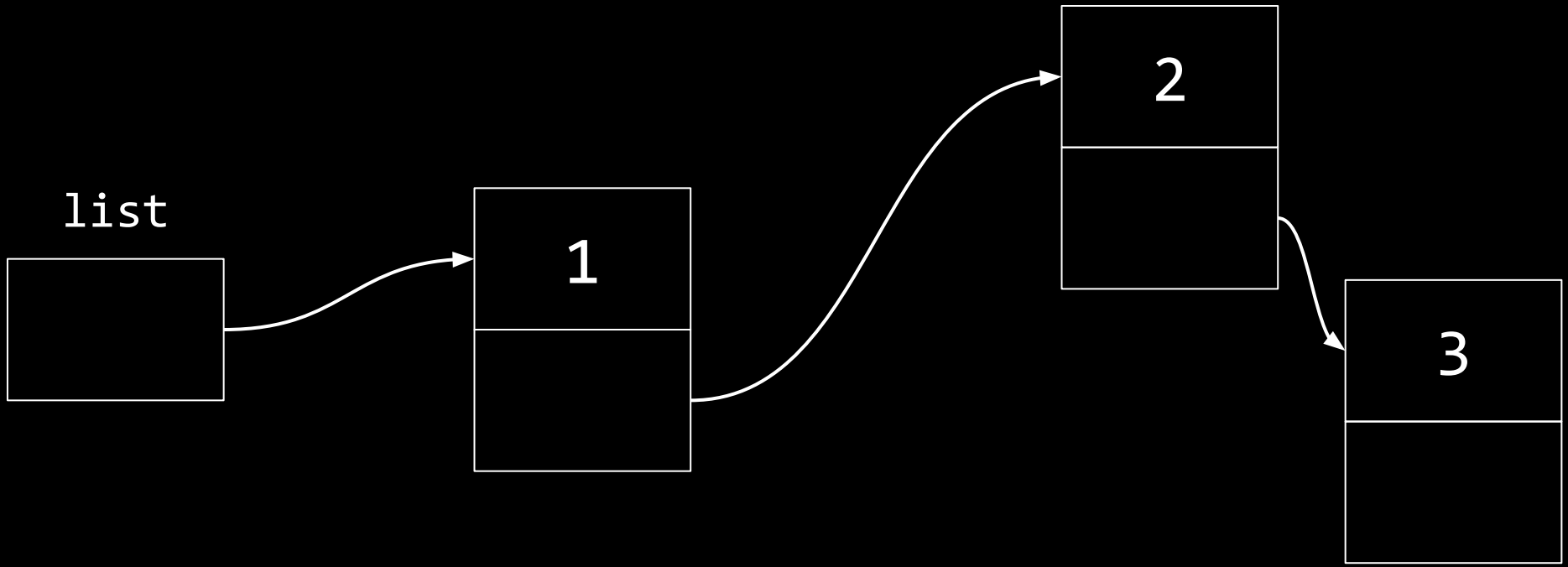

list
1
2

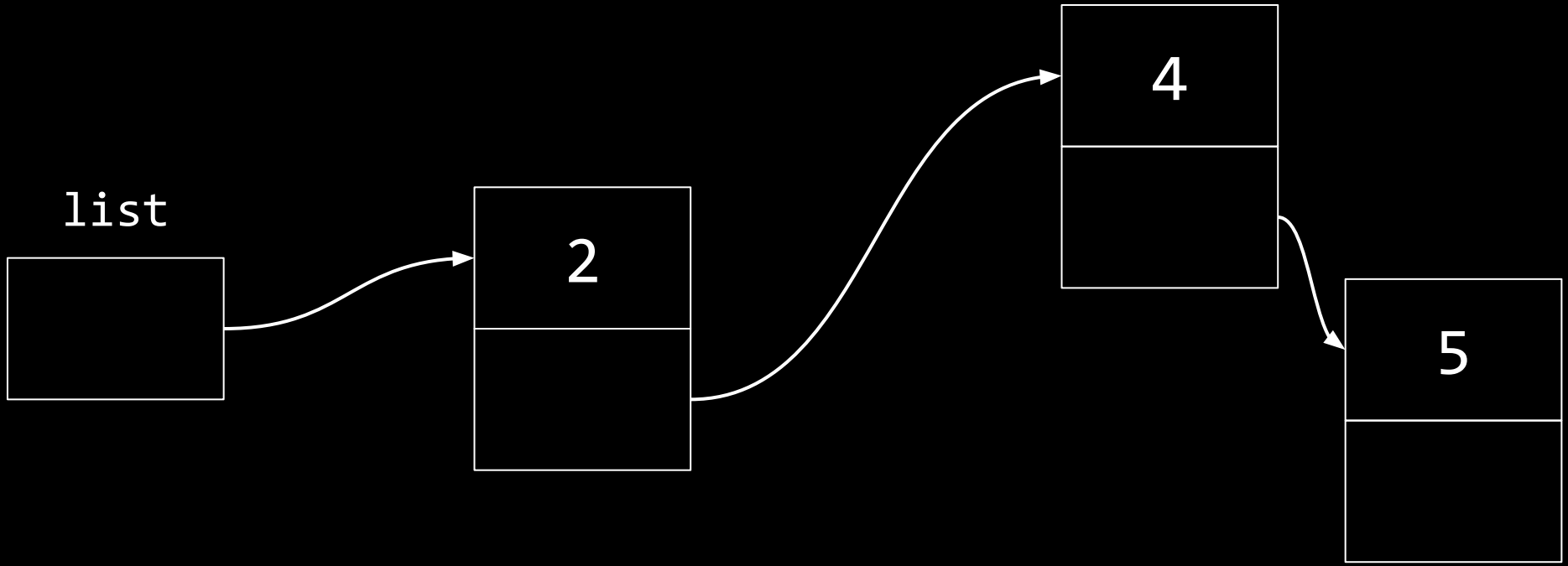
```
node *n = malloc(sizeof(node));
if (n != NULL)
{
    n->number = 3;
    n->next = NULL;
}
```


list
1
2
3
n

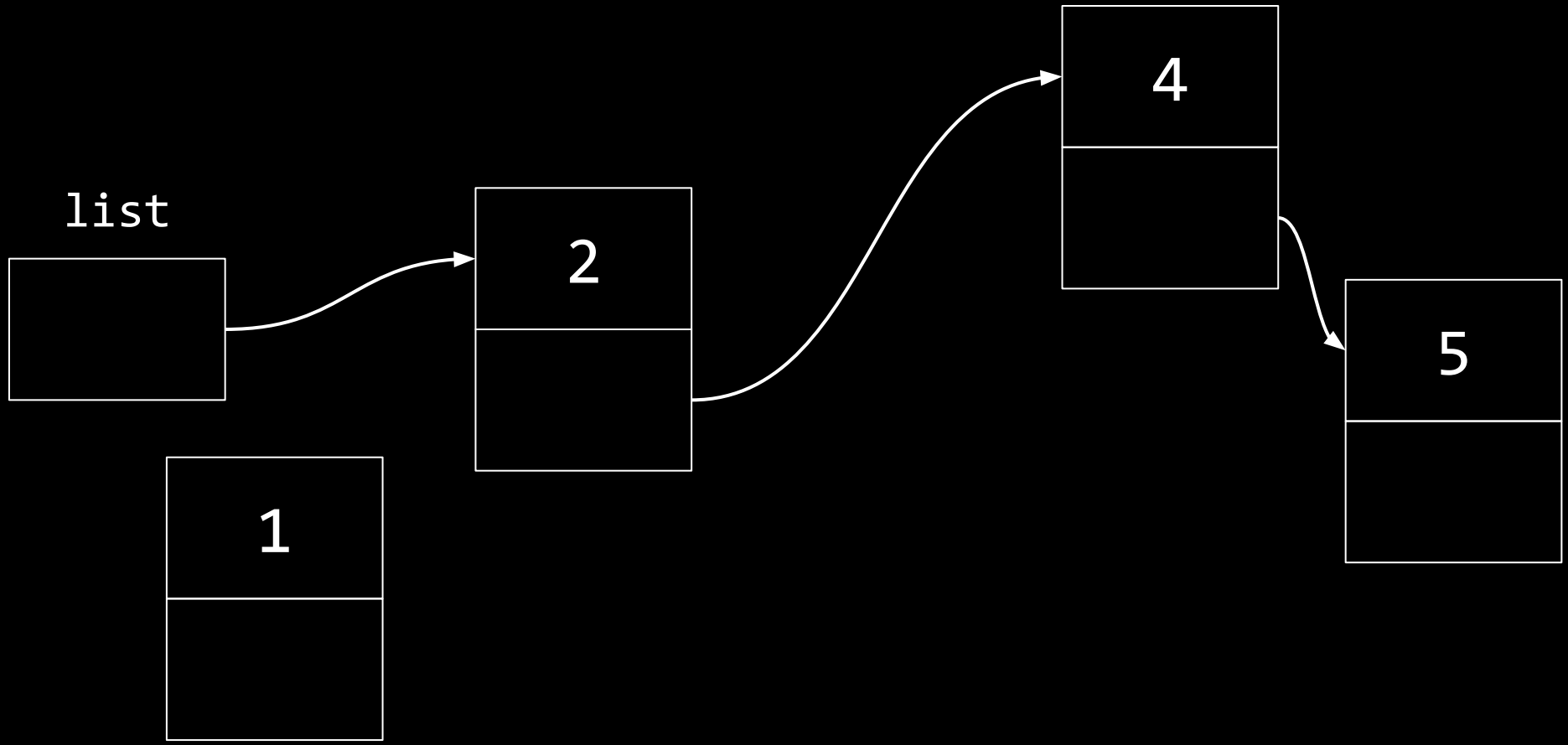
```
list->next->next = n;
```


list
1
2
3
n

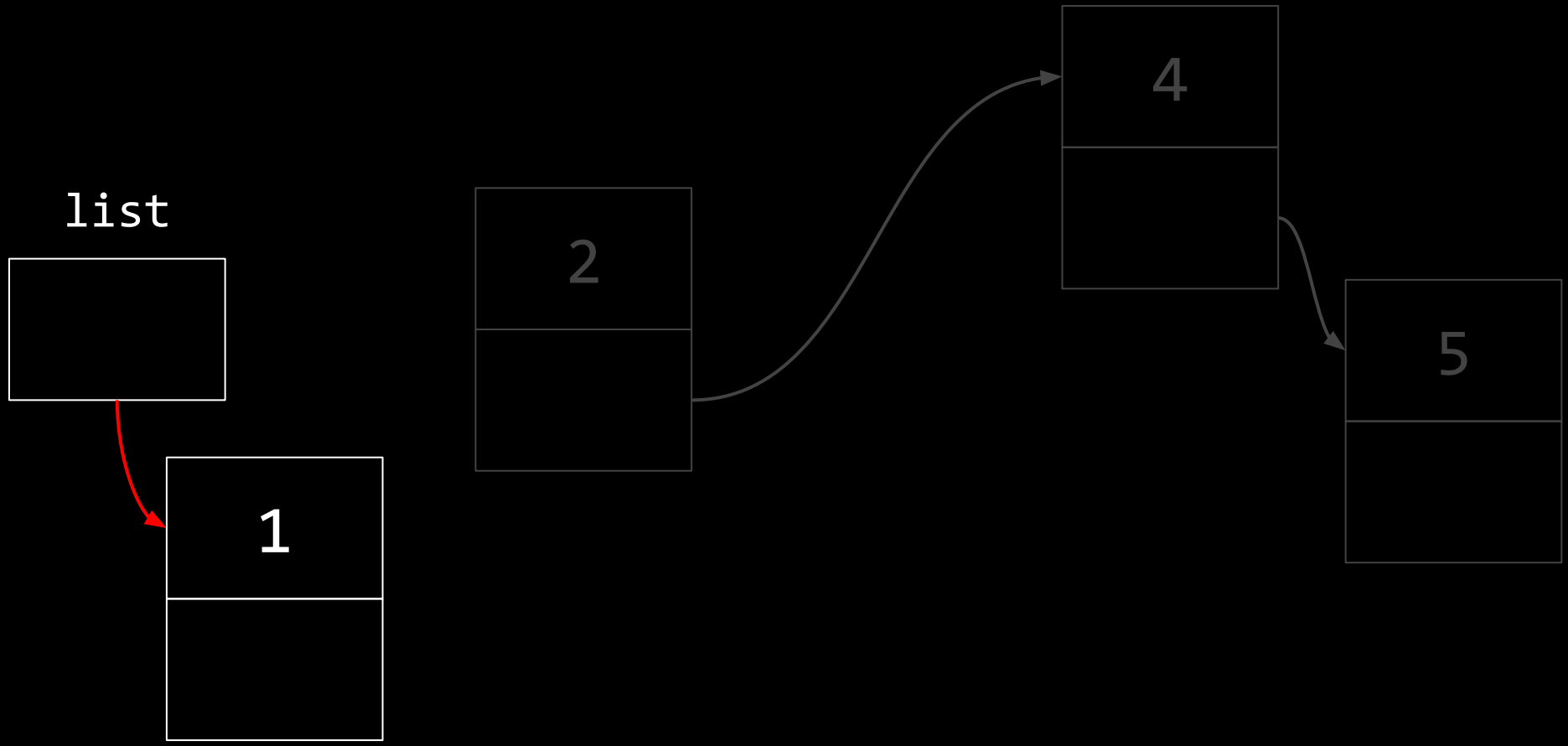

list
1
2
3


list
2
4
5

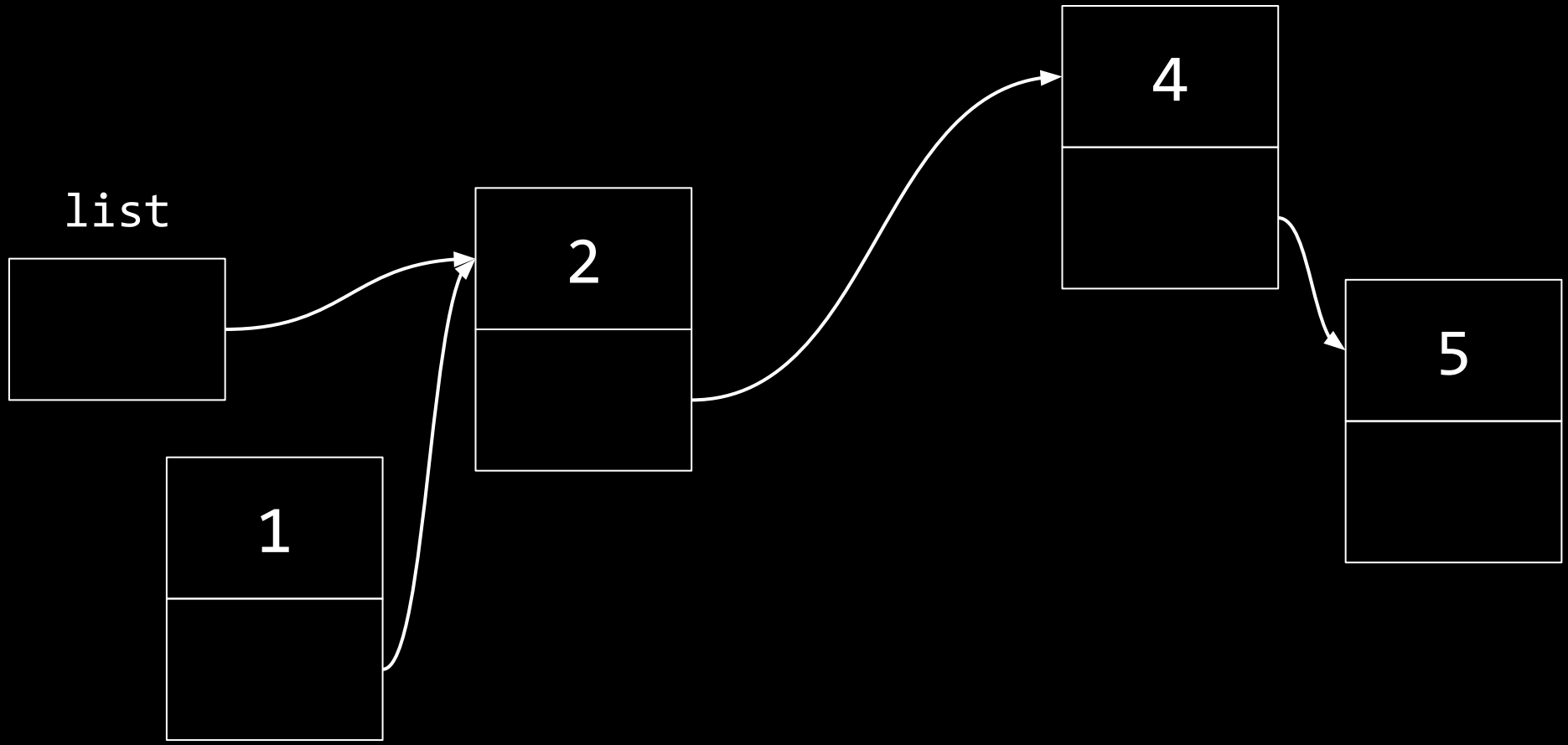
```
node *n = malloc(sizeof(node));
if (n != NULL)
{
    n->number = 1;
    n->next = NULL;
}
```


list
2
4
5
1

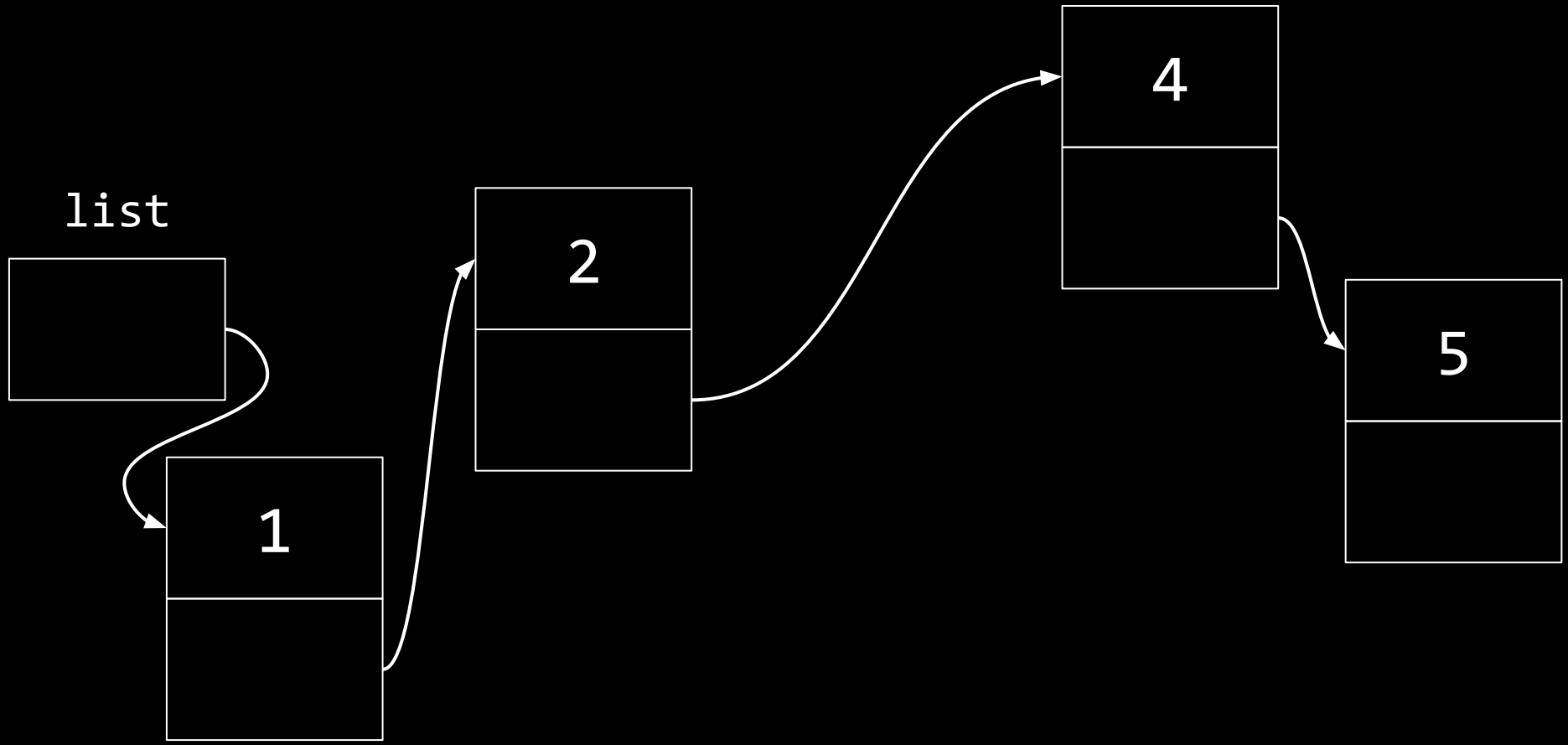
```
list = n;
```

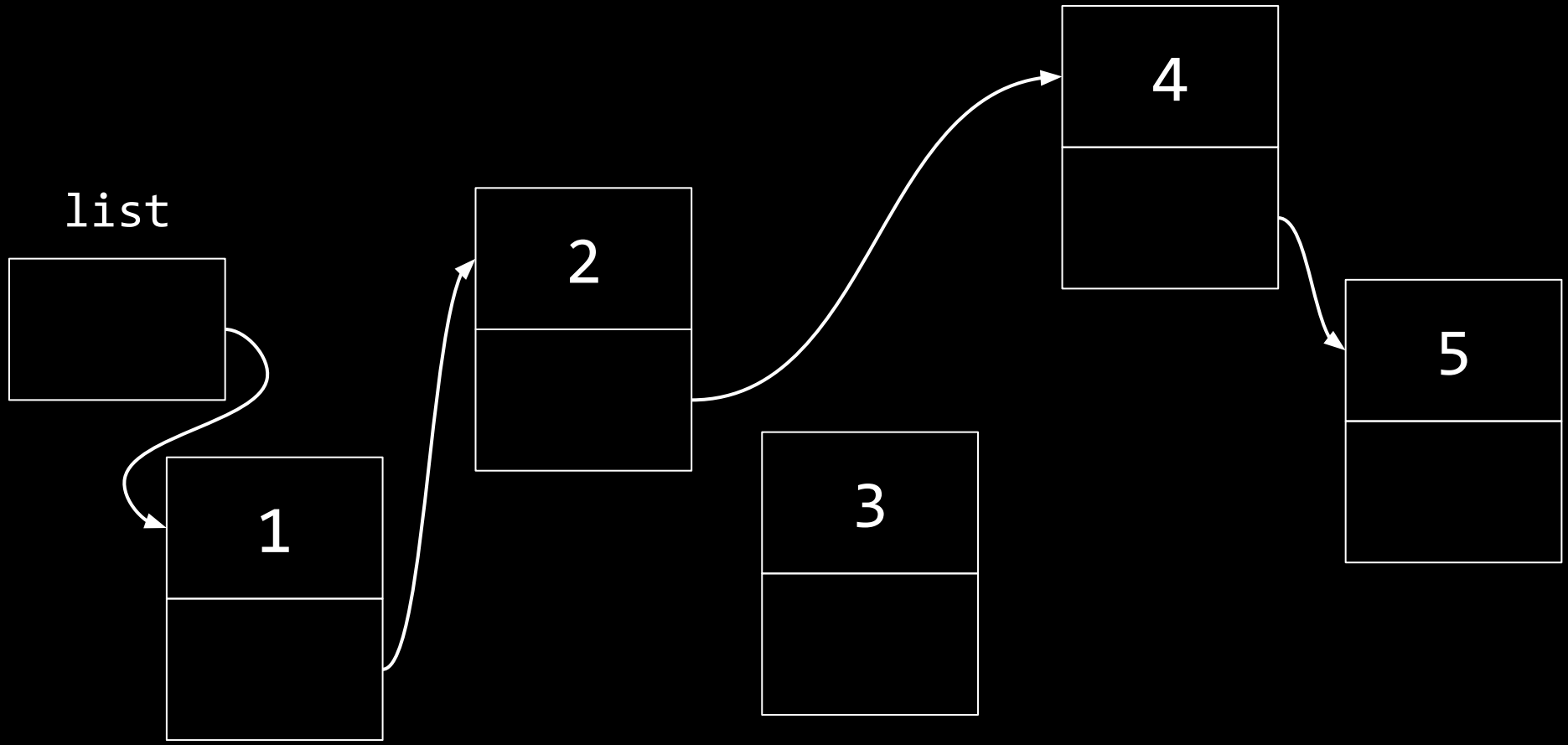

list
1
2
4
5

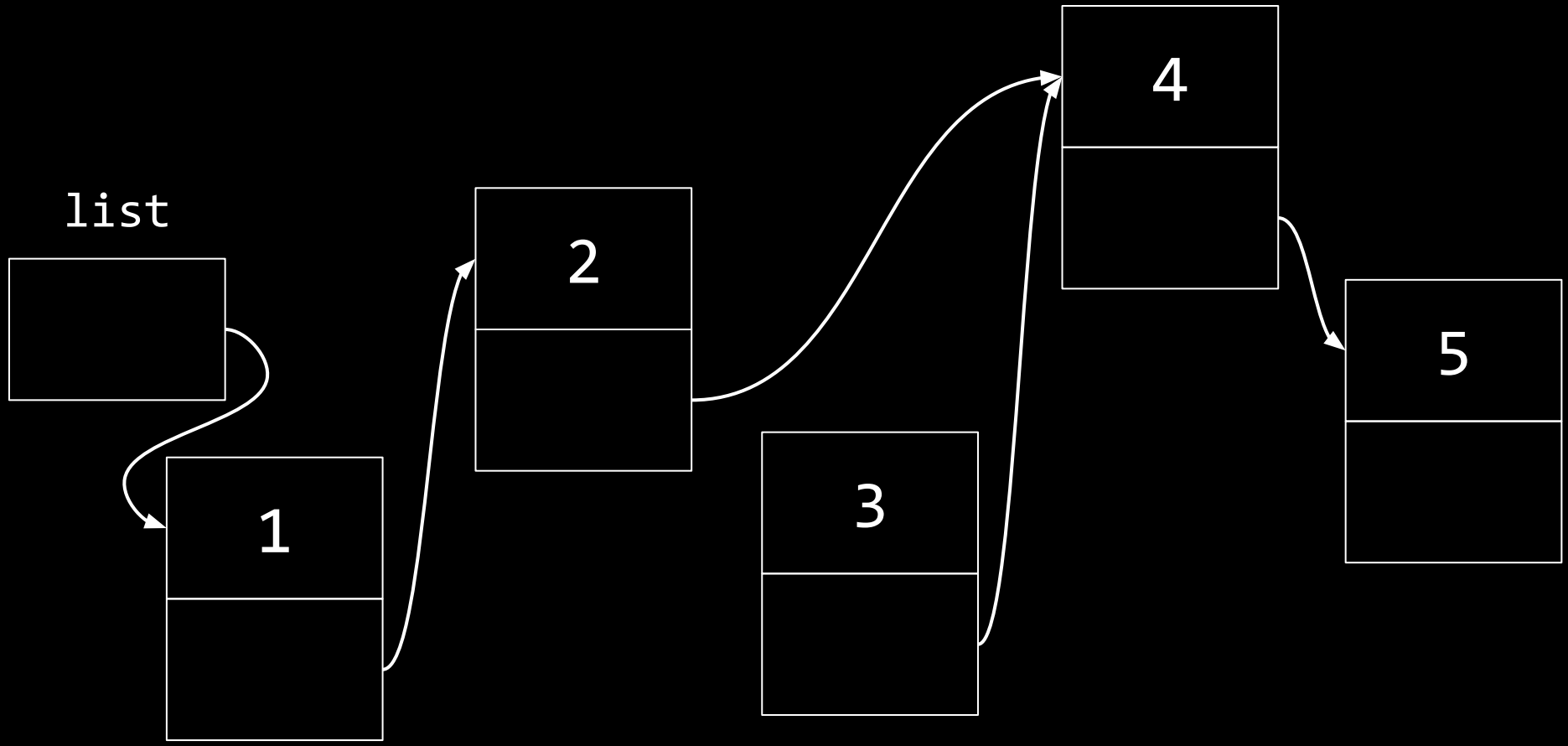
```
n->next = list;
```

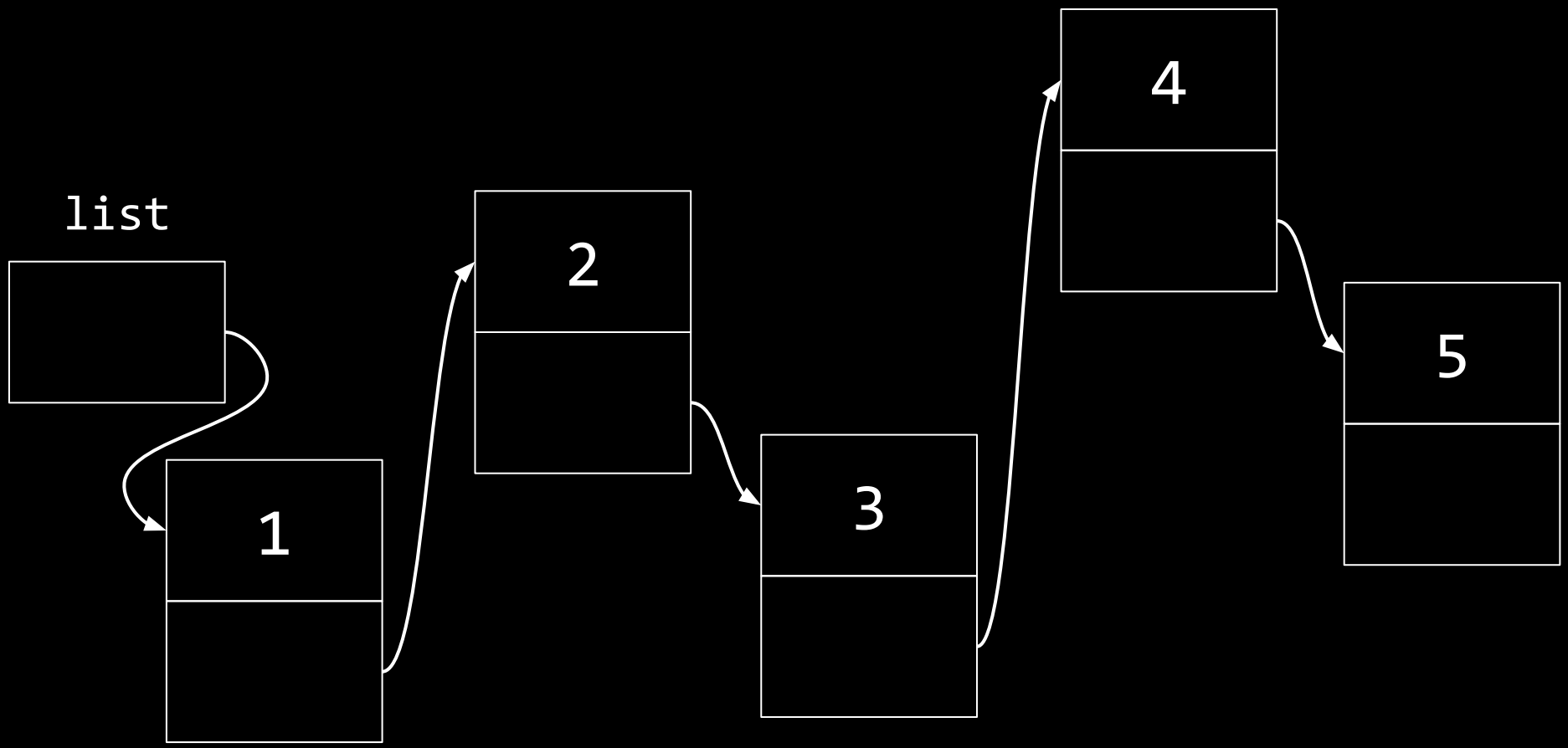

list
2
4
5
1

```
list = n;
```


list
1
2
4
5

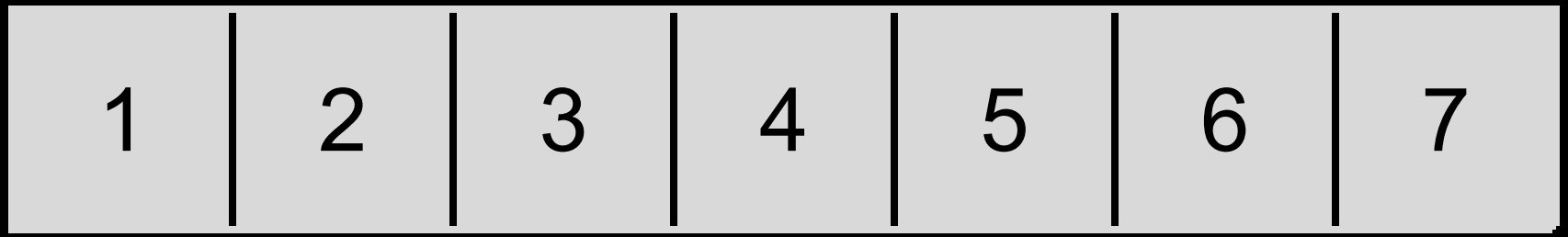

list
1
2
3
4
5

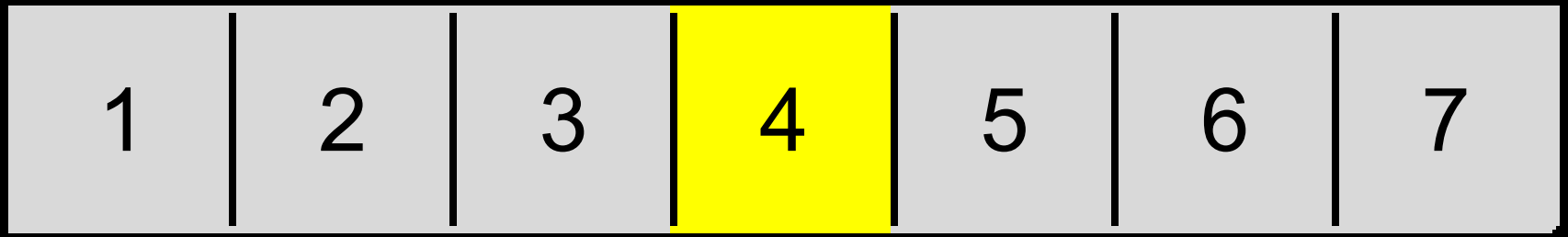

list
1
2
3
4
5

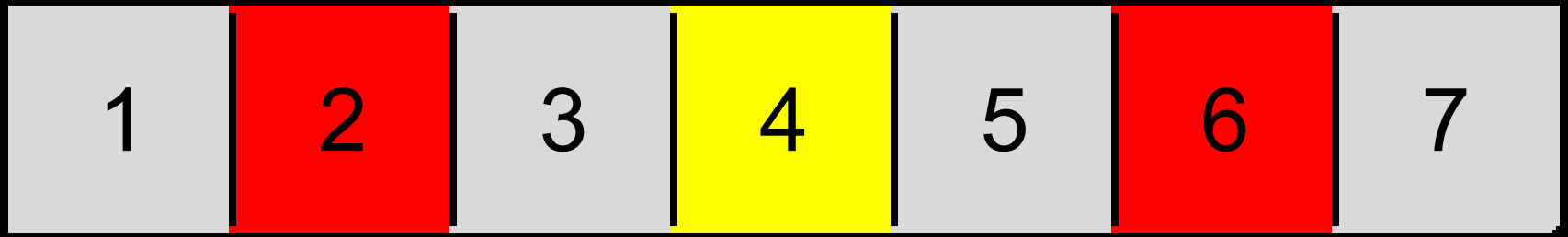

list
1
2
3
4
5

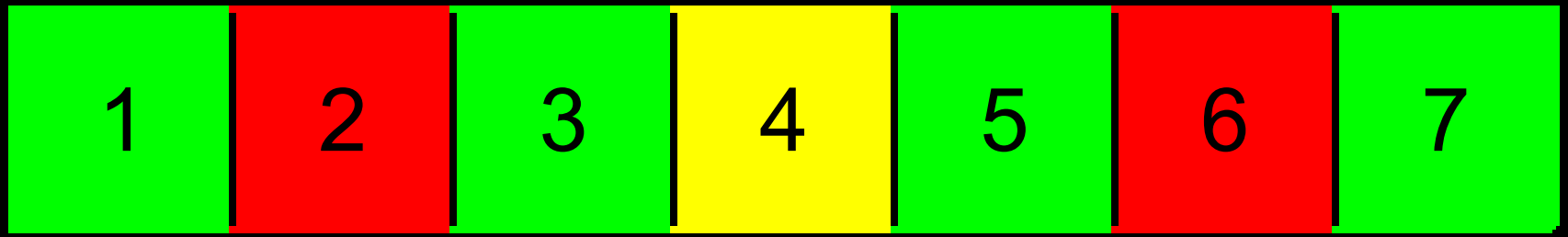
trees

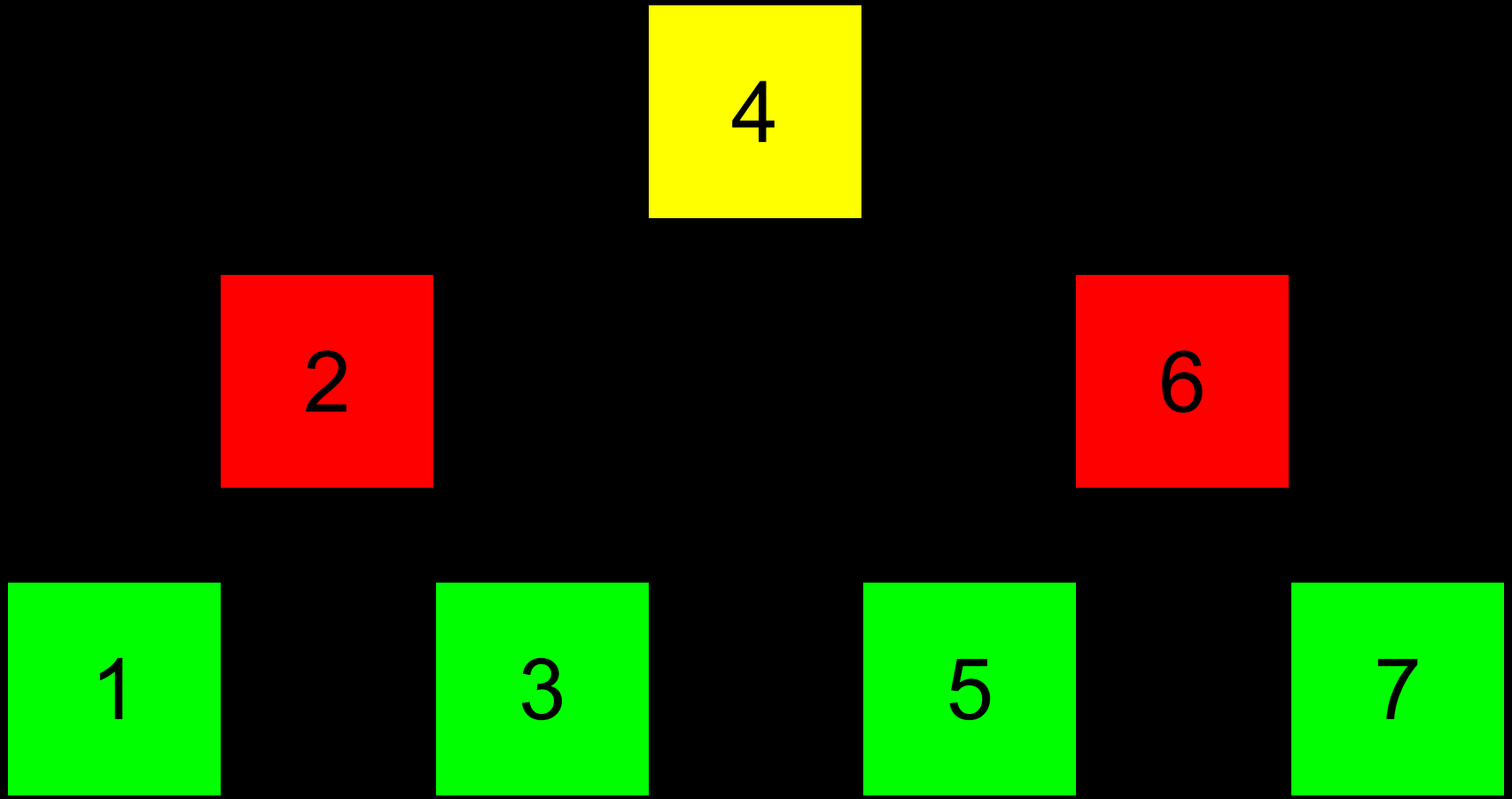
binary search trees

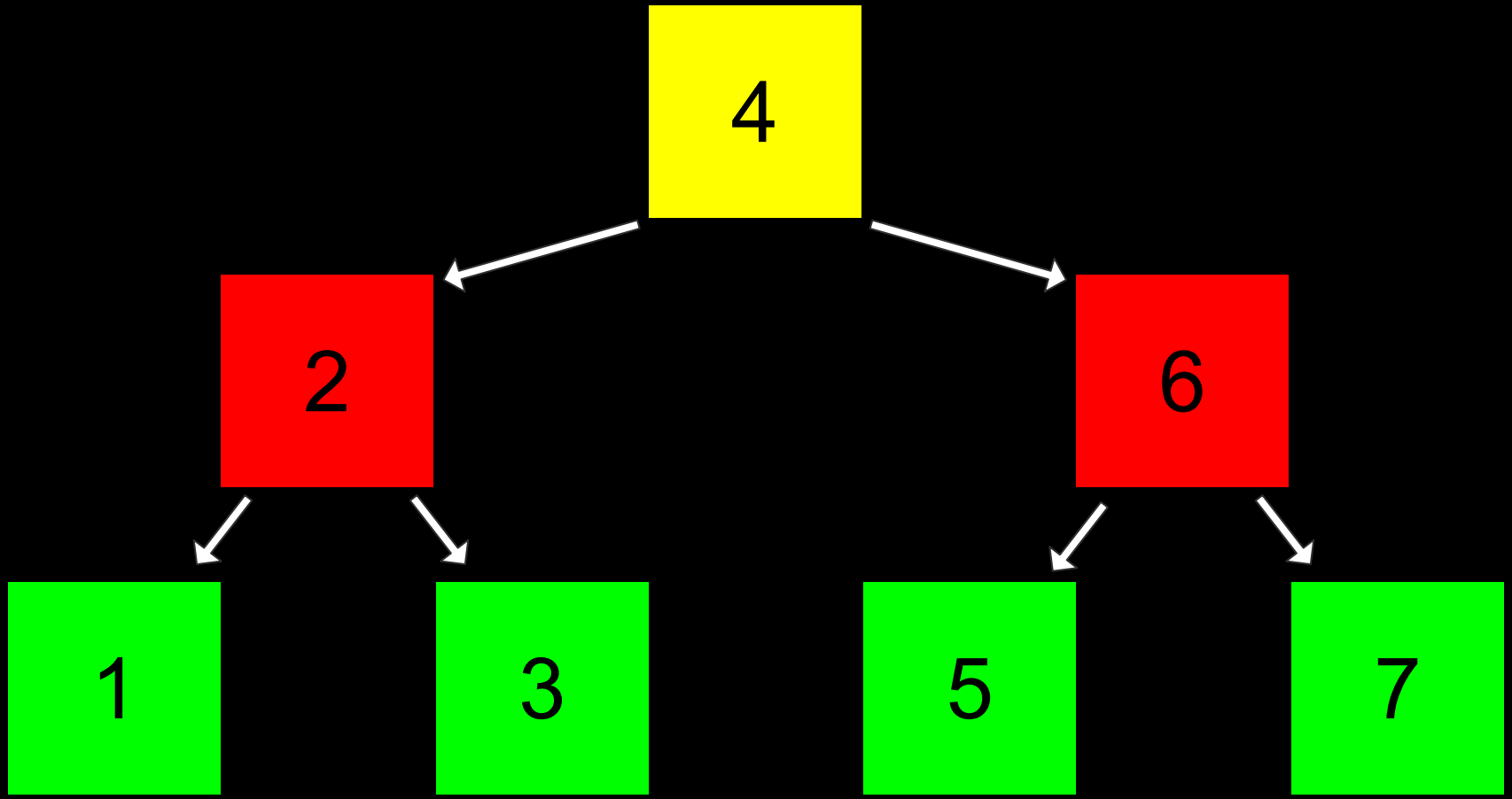

1
2
3
4
5
6
7


1
2
3
4
5
6
7


1
2
3
4
5
6
7


1
2
3
4
5
6
7


4
2
6
1
3
5
7

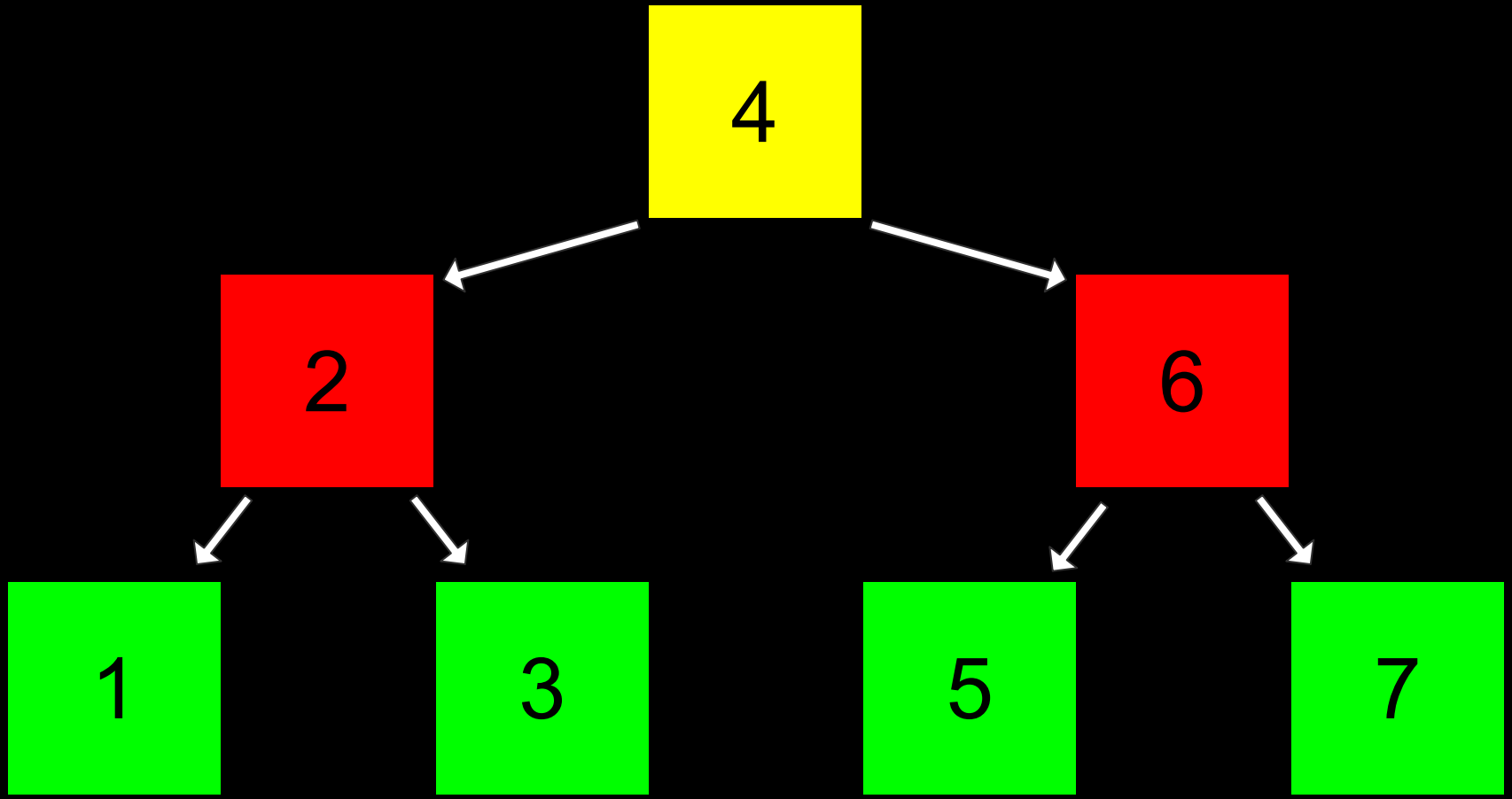

4
2
6
1
3
5
7

```
typedef struct node
{
    int number;
    struct node *next;
}
node;
```

```
typedef struct node
{
    int number;

}
node;
```

```
typedef struct node
{
    int number;


}
node;
```

```
typedef struct node
{
    int number;
    struct node *left;
    struct node *right;
}
node;
```


4
2
6
1
3
5
7

```
bool search(node *tree, int number)
{









}
```

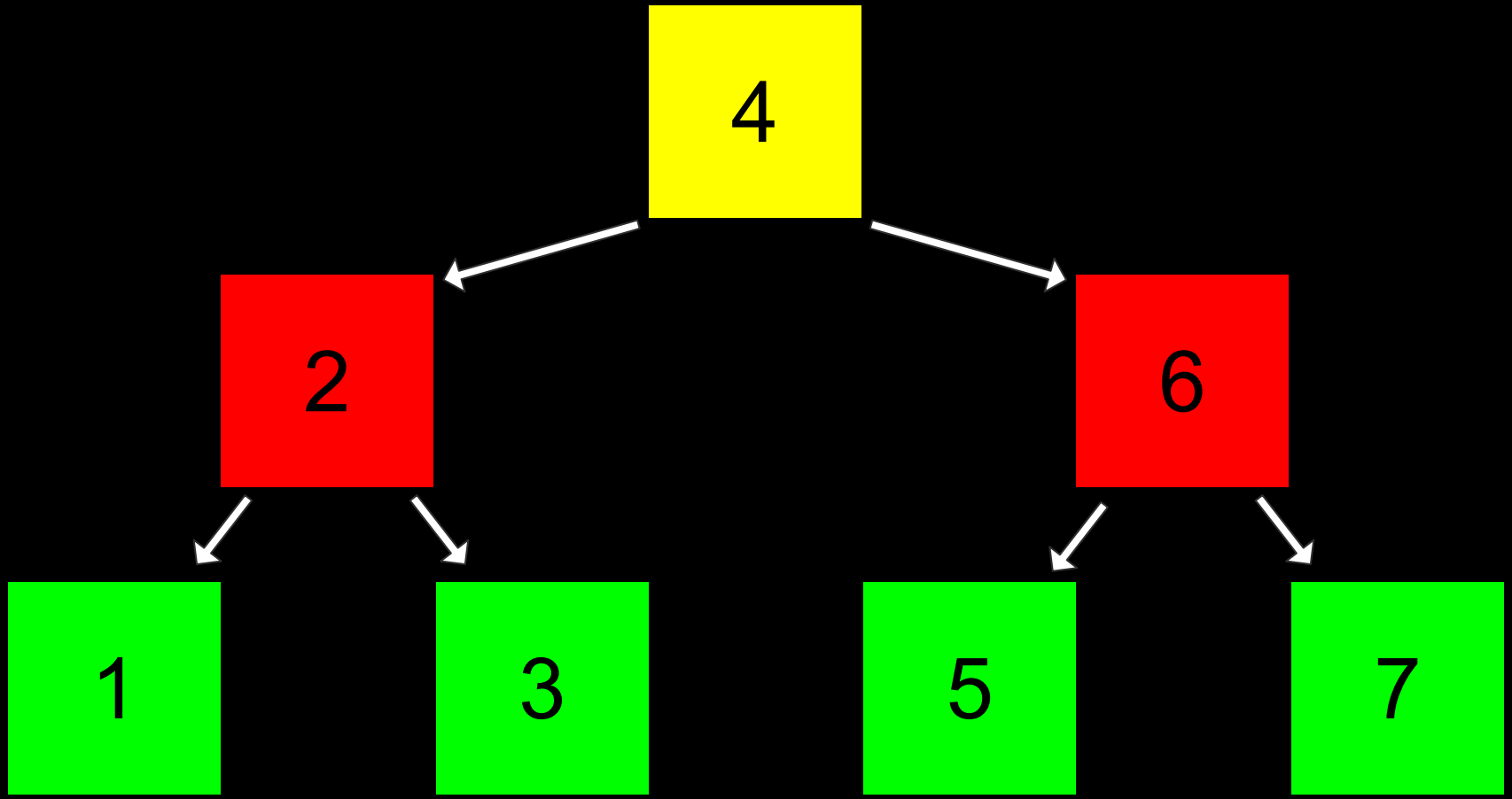
```
bool search(node *tree, int number)
{
    if (tree == NULL)
    {
        return false;
    }









}
```

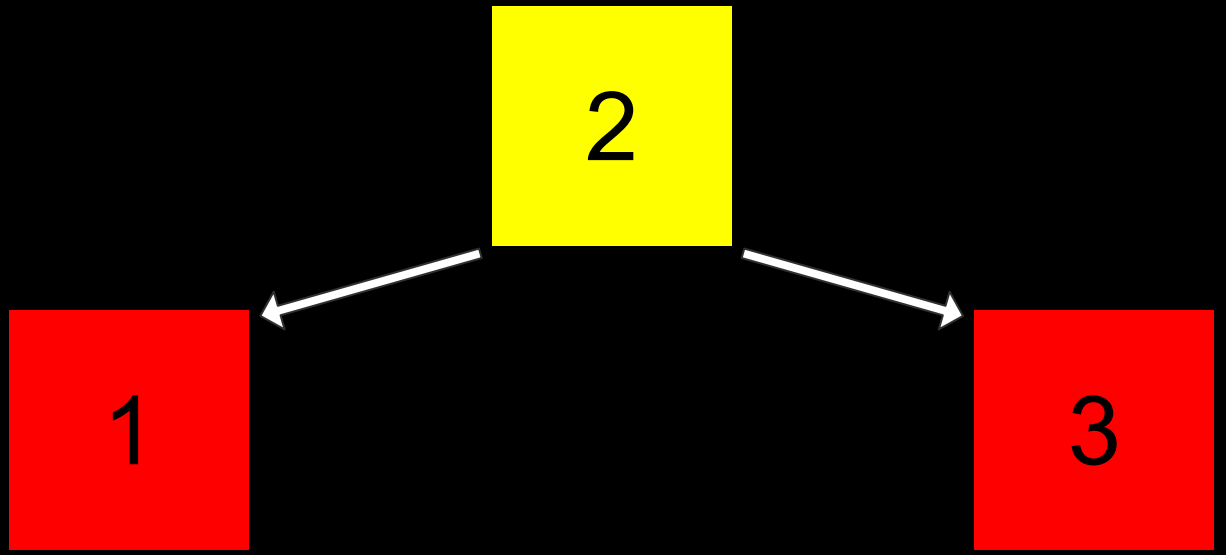
```
bool search(node *tree, int number)
{
    if (tree == NULL)
    {
        return false;
    }
    else if (number < tree->number)
    {
        return search(tree->left, number);
    }









}
```

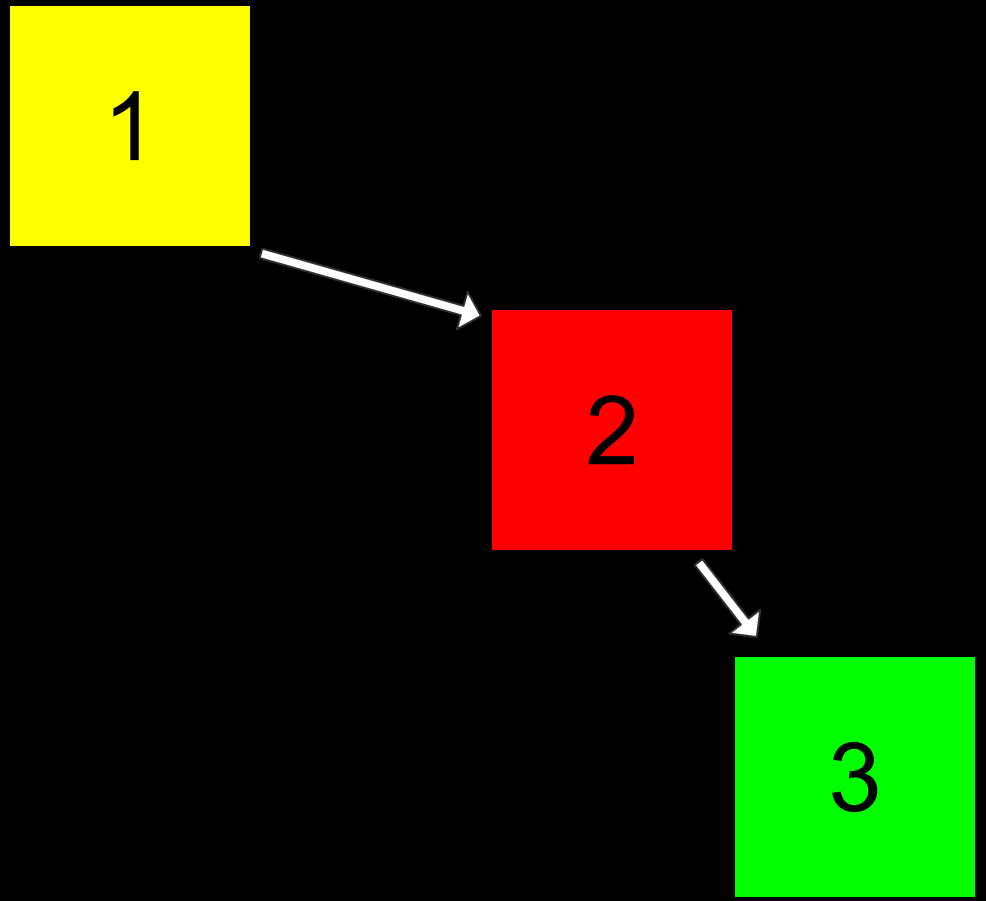
```
bool search(node *tree, int number)
{
    if (tree == NULL)
    {
        return false;
    }
    else if (number < tree->number)
    {
        return search(tree->left, number);
    }
    else if (number > tree->number)
    {
        return search(tree->right, number);
    }




}
```

```
bool search(node *tree, int number)
{
    if (tree == NULL)
    {
        return false;
    }
    else if (number < tree->number)
    {
        return search(tree->left, number);
    }
    else if (number > tree->number)
    {
        return search(tree->right, number);
    }
    else if (number == tree->number)
    {
        return true;
    }
}
```

```
bool search(node *tree, int number)
{
    if (tree == NULL)
    {
        return false;
    }
    else if (number < tree->number)
    {
        return search(tree->left, number);
    }
    else if (number > tree->number)
    {
        return search(tree->right, number);
    }
    else
    {
        return true;
    }
}
```


4
2
6
1
3
5
7


2
1
3


1
2
3

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$ search

$O(1)$

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$ search, insert

$O(1)$

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

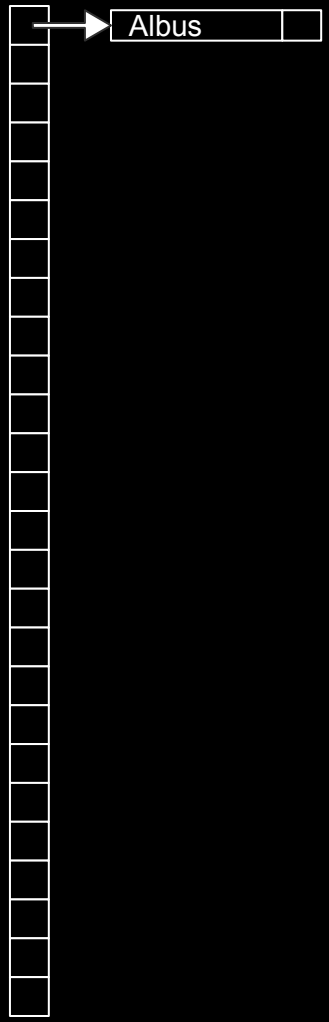
$\Omega(\log n)$

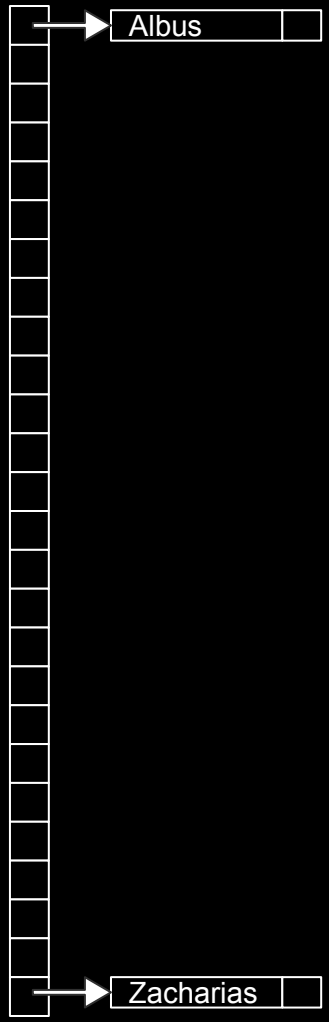
$\Omega(1)$

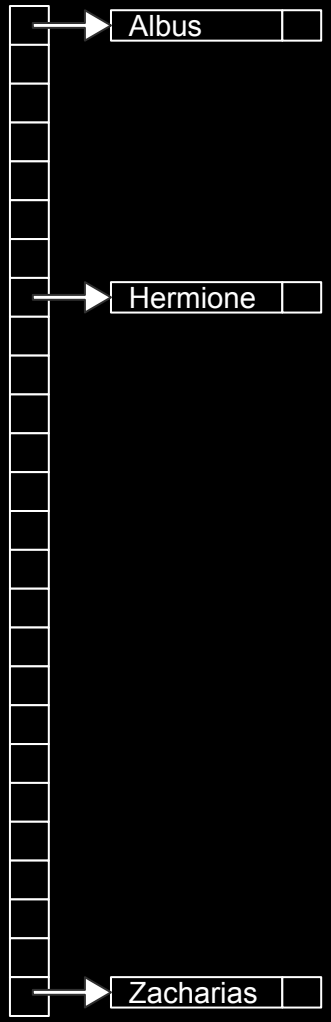
hash tables

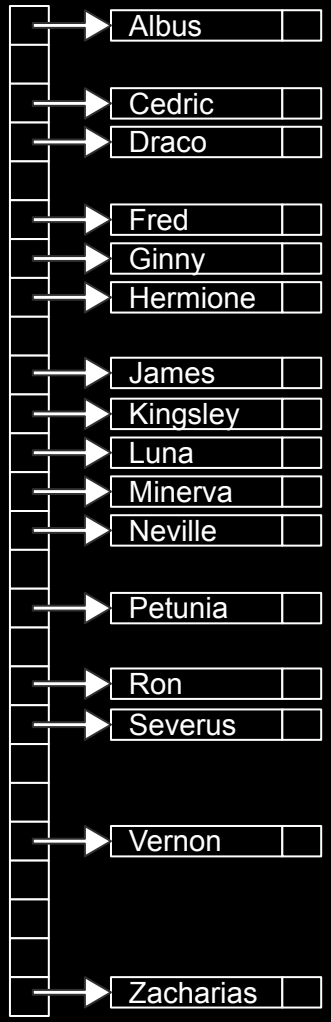

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25

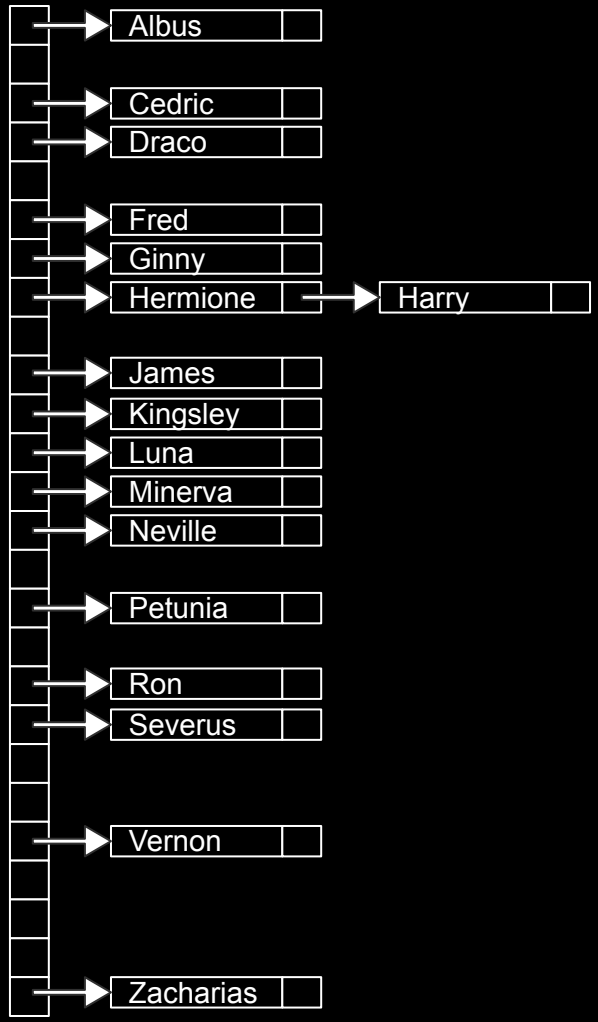

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z


Albus


Albus
Zacharias

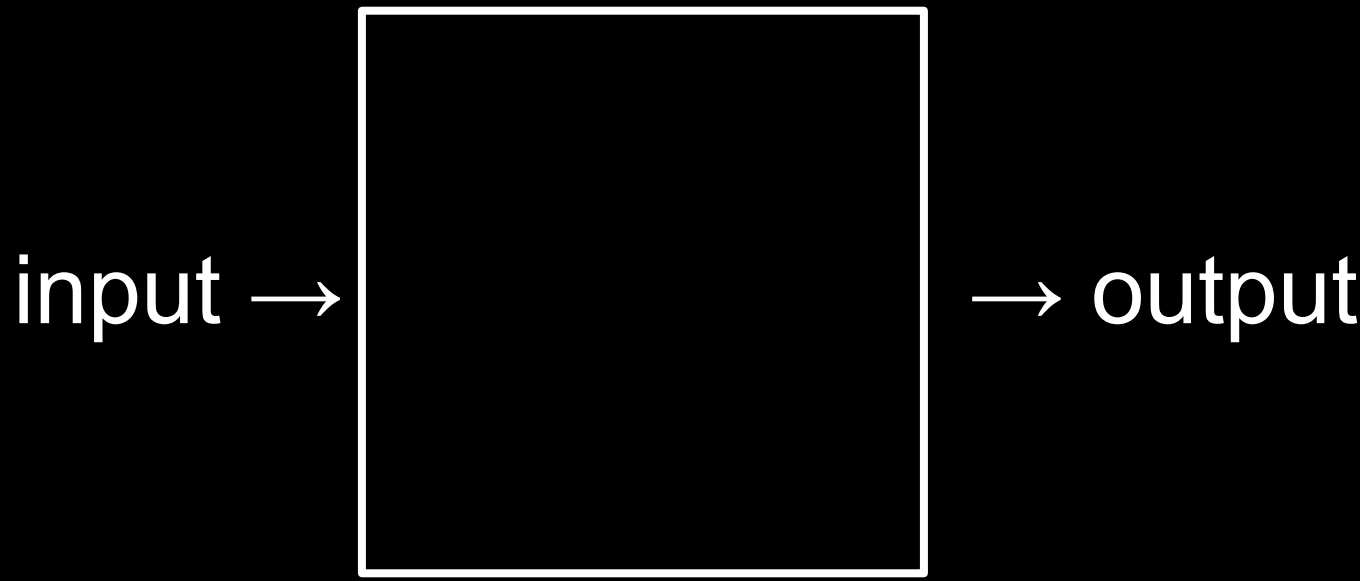

Albus
Hermione
Zacharias

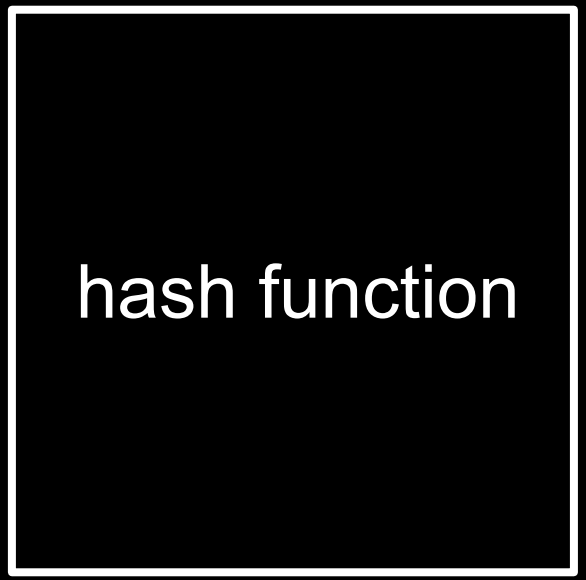

Albus
Cedric
Draco
Fred
Ginny
Hermione
James
Kingsley
Luna
Minerva
Neville
Petunia
Ron
Severus
Vernon
Zacharias

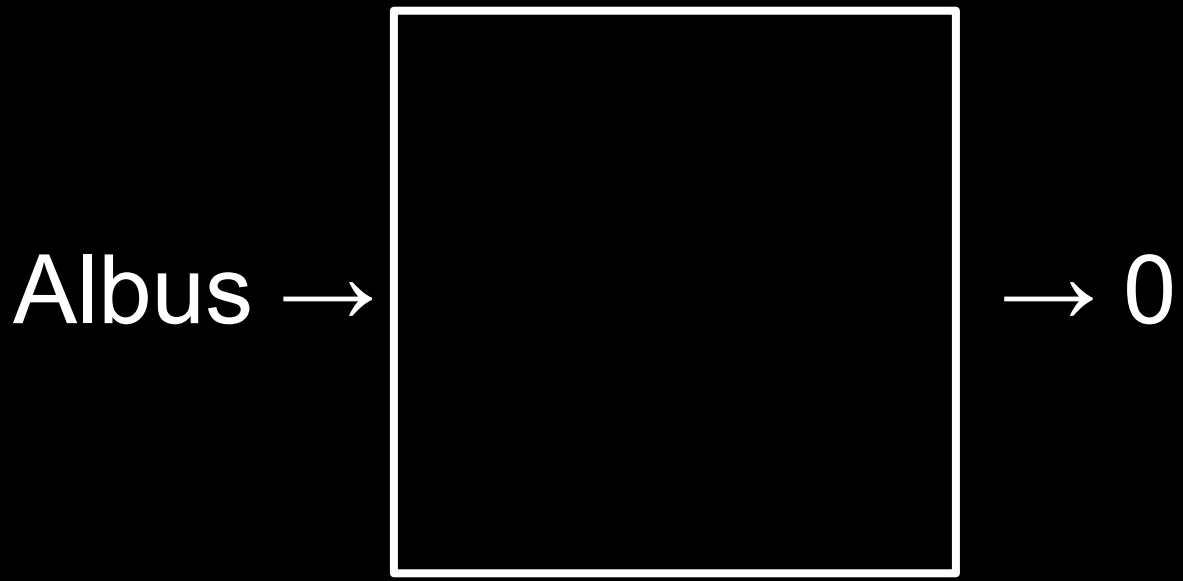

Albus
Cedric
Draco
Fred
Ginny
Hermione
Harry
James
Kingsley
Luna
Minerva
Neville
Petunia
Ron
Severus
Vernon
Zacharias

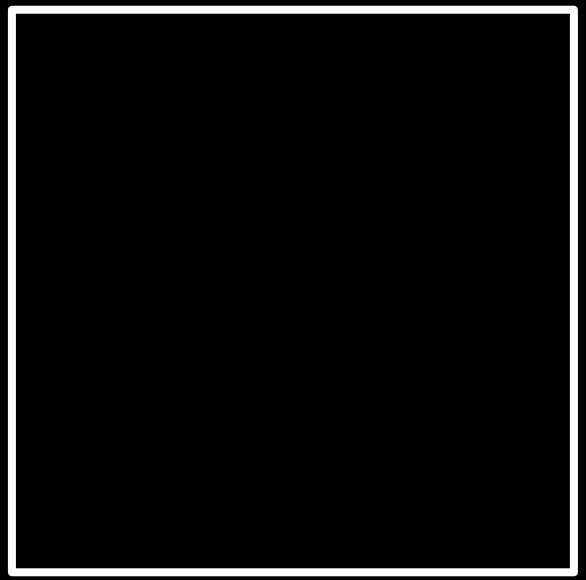

Albus
Cedric
Draco
Fred
Ginny
Hermione
Harry
Hagrid
James
Kingsley
Luna
Minerva
Neville
Petunia
Ron
Severus
Vernon
Zacharias

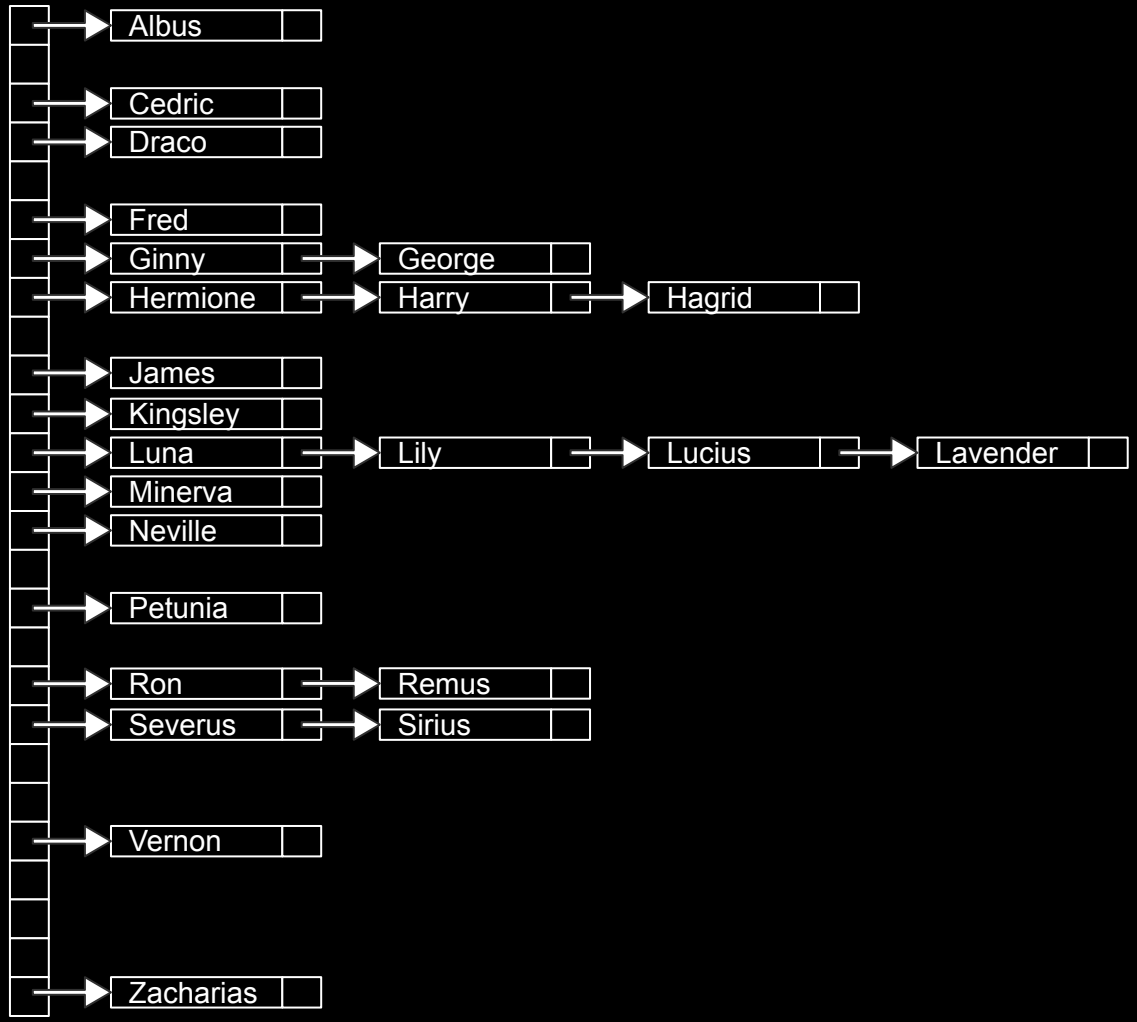

Albus
Cedric
Draco
Fred
Ginny
George
Hermione
Harry
Hagrid
James
Kingsley
Luna
Lily
Lucius
Minerva
Neville
Petunia
Ron
Remus
Severus
Sirius
Vernon
Zacharias

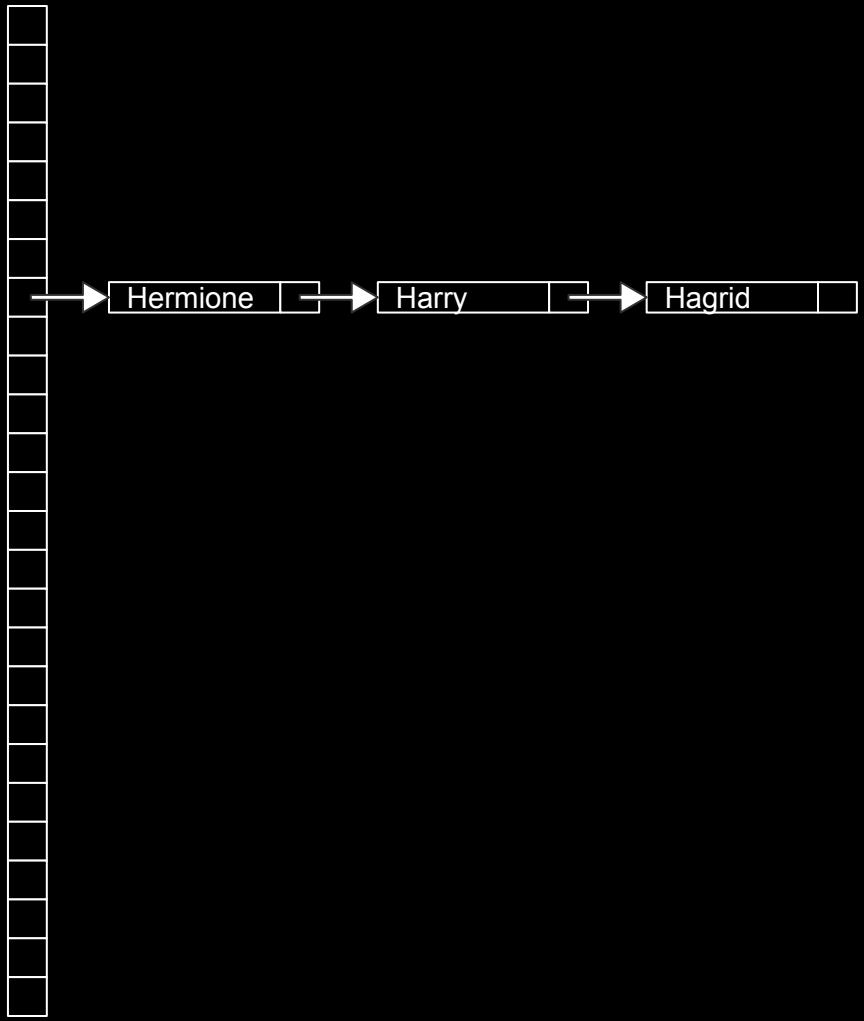

input →
→ output

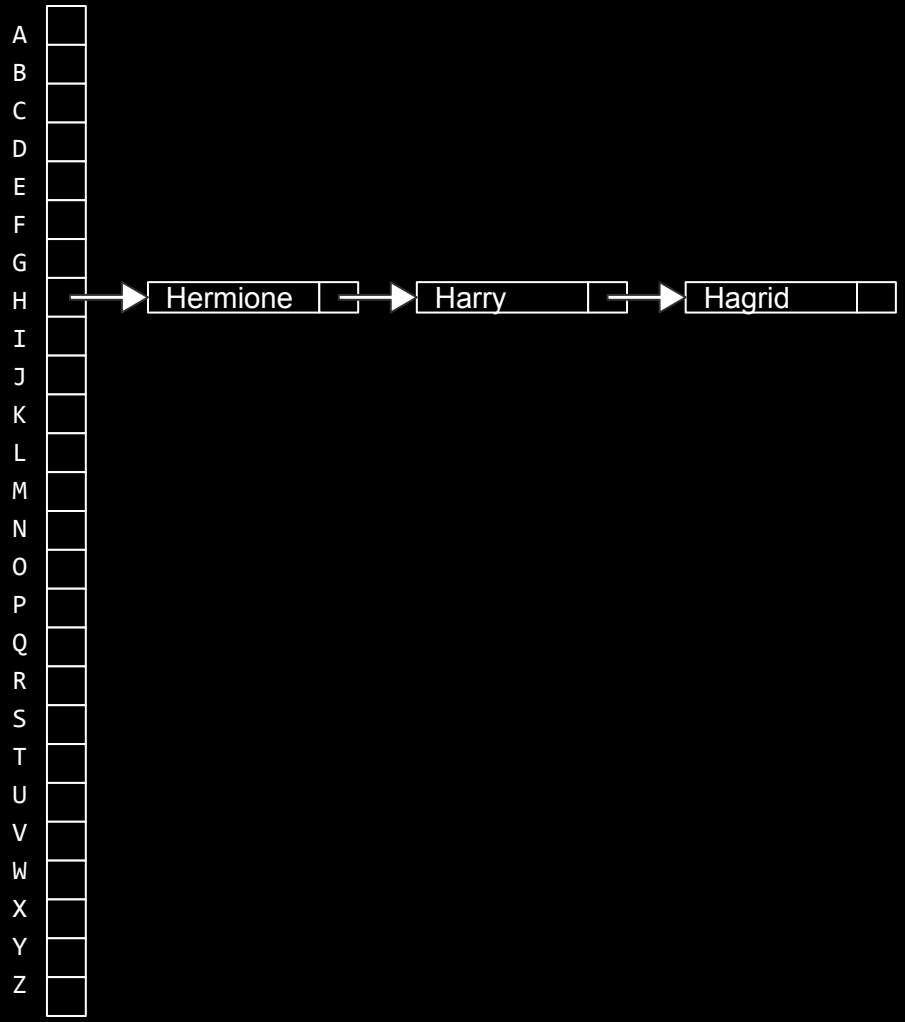

hash function

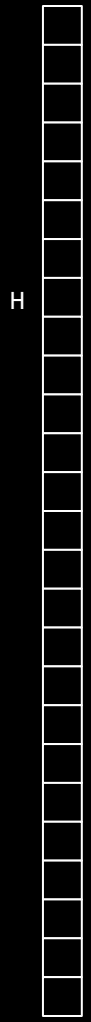

Albus →
→ 0

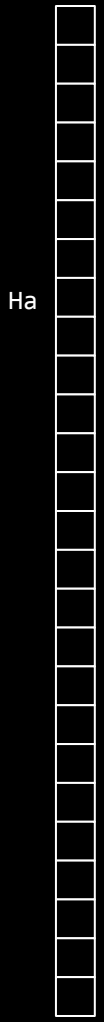
Zacharias →  → 25


Albus
Cedric
Draco
Fred
Ginny
George
Hermione
Harry
Hagrid
James
Kingsley
Luna
Lily
Lucius
Lavender
Minerva
Neville
Petunia
Ron
Remus
Severus
Sirius
Vernon
Zacharias

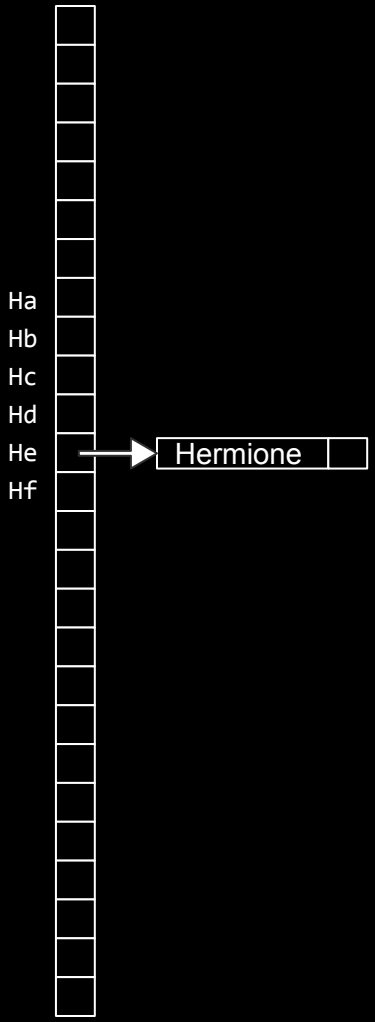

Hermione
Harry
Hagrid

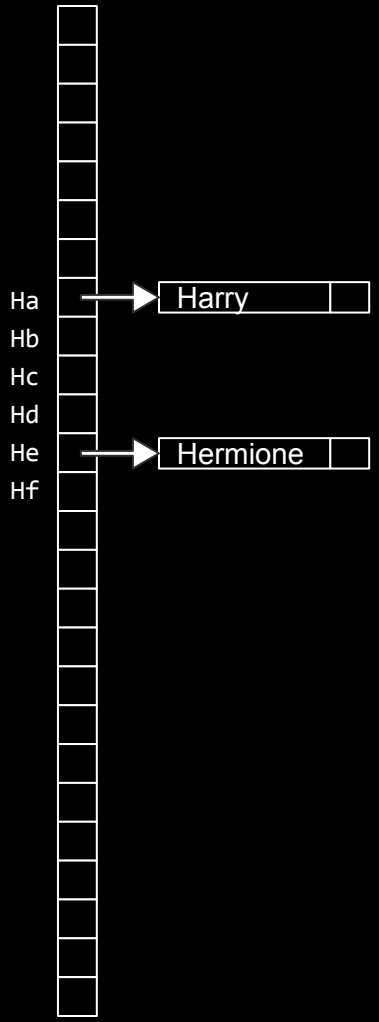

A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
Hermione
Harry
Hagrid

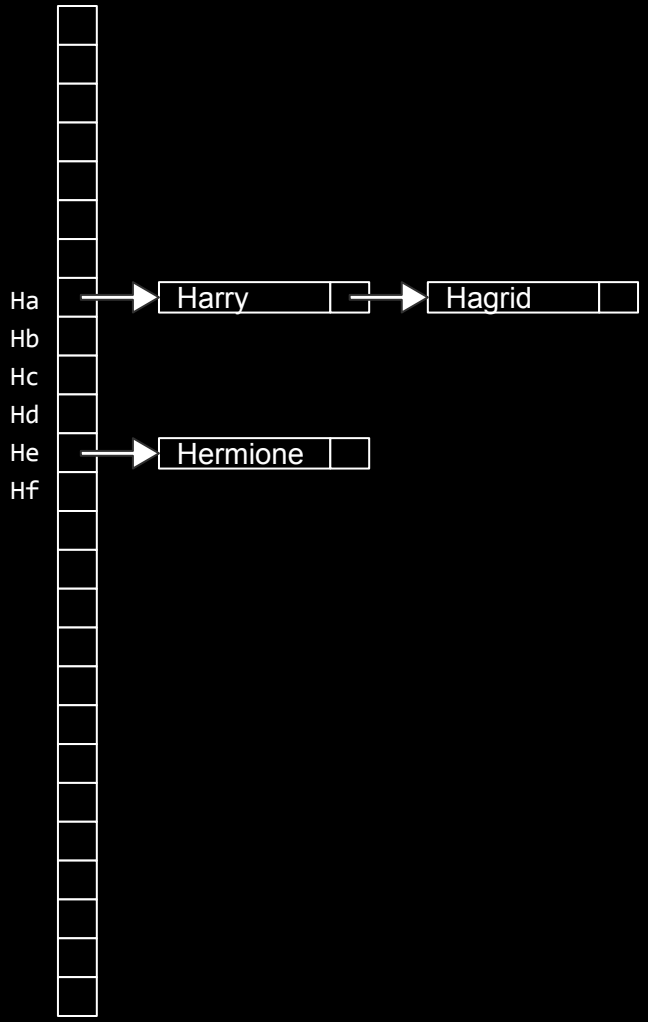

H

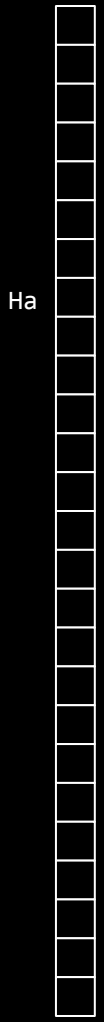

Ha


Ha
Hb
Hc
Hd
He
Hf


Ha
Hb
Hc
Hd
He
Hf
Hermione

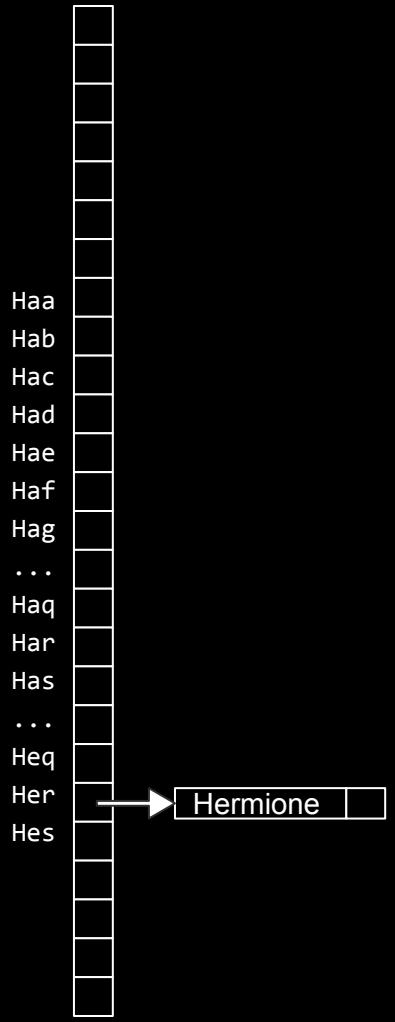

Ha
Hb
Hc
Hd
He
Hf
Harry
Hermione

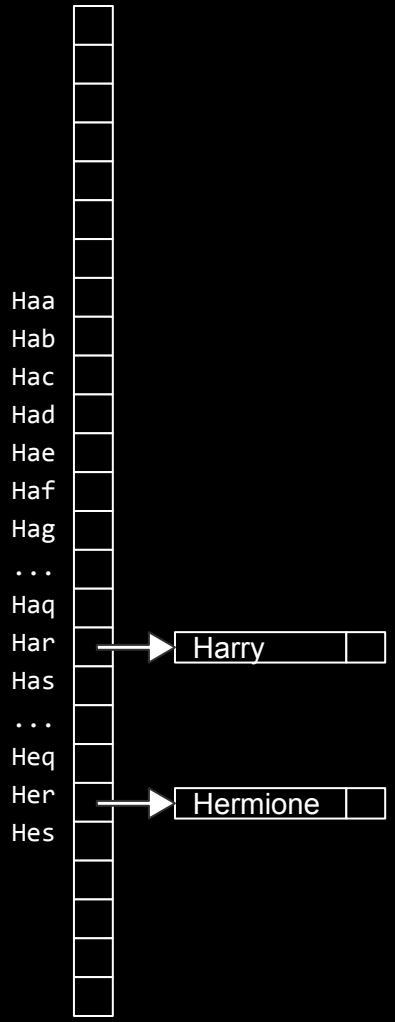

Ha
Hb
Hc
Hd
He
Hf
Harry
Hagrid
Hermione

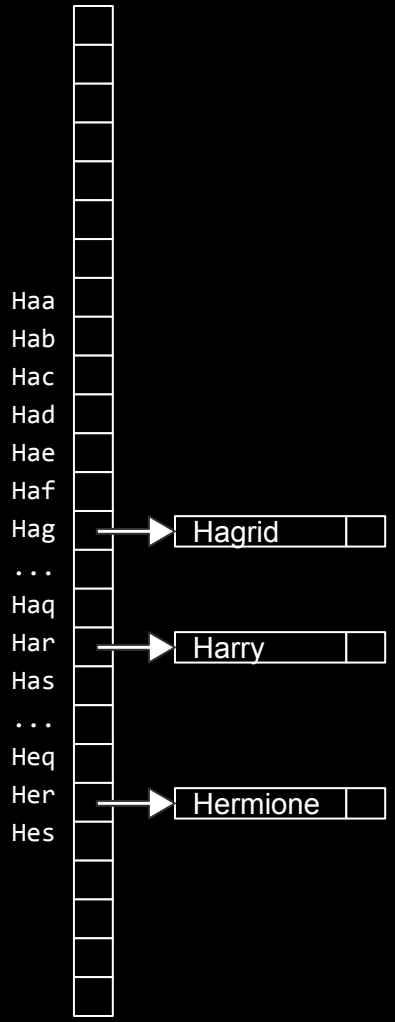

Ha

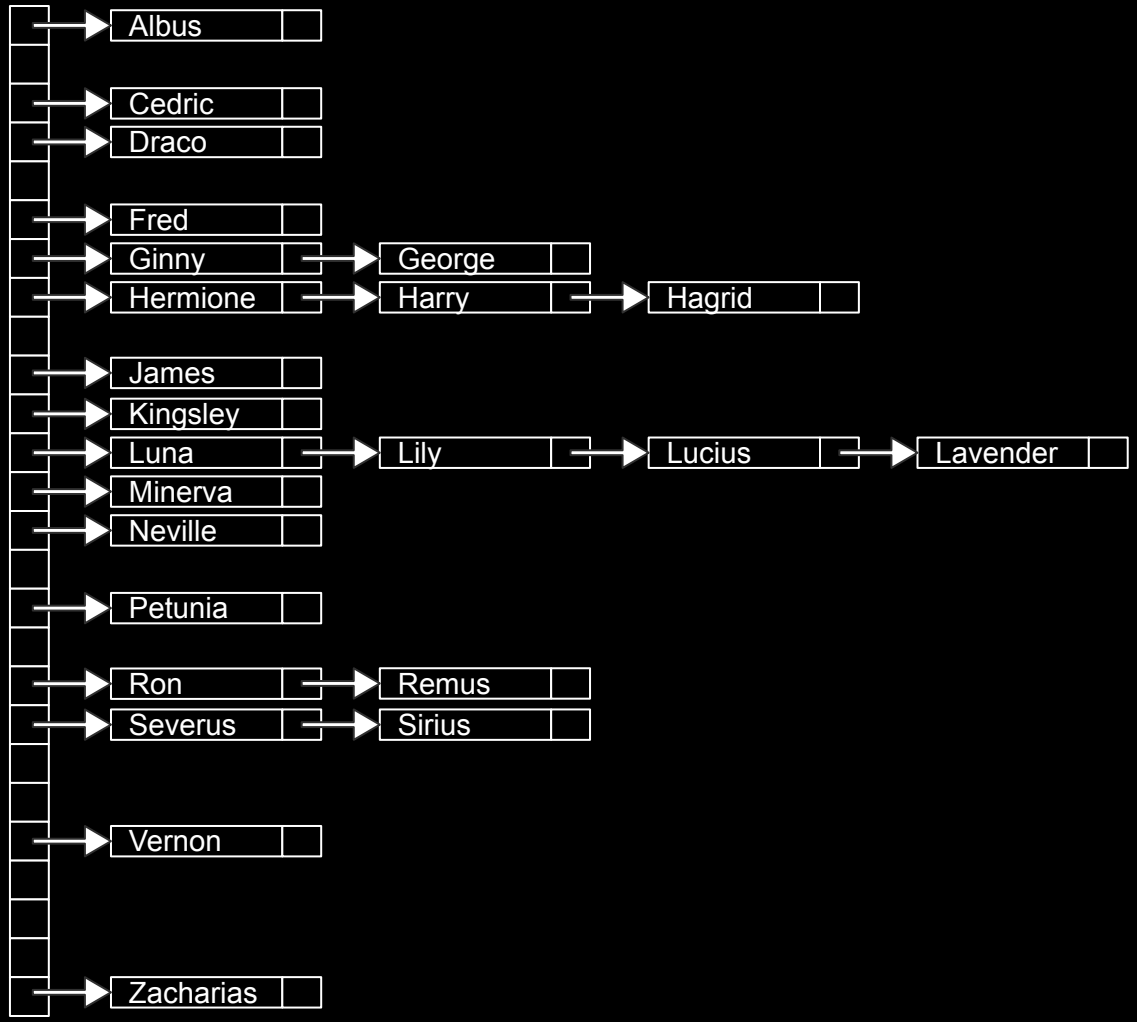
Haa


Haa
Hab
Hac
Had
Hae
Haf
Hag
...
Haq
Har
Has
...
Heq
Her
Hes


Haa
Hab
Hac
Had
Hae
Haf
Hag
...
Haq
Har
Has
...
Heq
Her
Hes
Hermione


Haa
Hab
Hac
Had
Hae
Haf
Hag
...
Haq
Har
Has
...
Heq
Her
Hes
Harry
Hermione


Haa
Hab
Hac
Had
Hae
Haf
Hag
...
Haq
Har
Has
...
Heq
Her
Hes
Hagrid
Harry
Hermione


Albus
Cedric
Draco
Fred
Ginny
George
Hermione
Harry
Hagrid
James
Kingsley
Luna
Lily
Lucius
Lavender
Minerva
Neville
Petunia
Ron
Remus
Severus
Sirius
Vernon
Zacharias

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(n^2)$

$O(n \log n)$

$O(n)$ search

$O(\log n)$

$O(1)$

$O(n^2)$

$O(n \log n)$

$O(n)$ search, insert

$O(\log n)$

$O(1)$

$O(n^2)$

$O(n \log n)$

$O(n)$ search

$O(\log n)$

$O(1)$ insert

$\Omega(n^2)$

$\Omega(n \log n)$

$\Omega(n)$

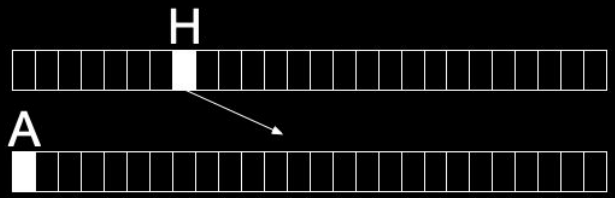
$\Omega(\log n)$

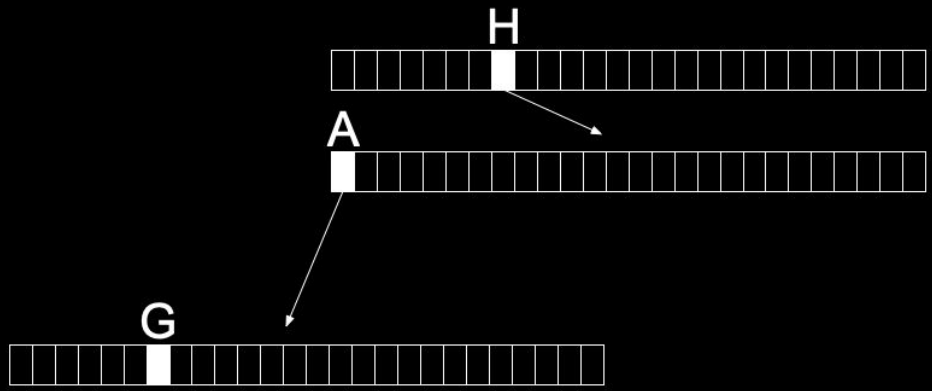
$\Omega(1)$

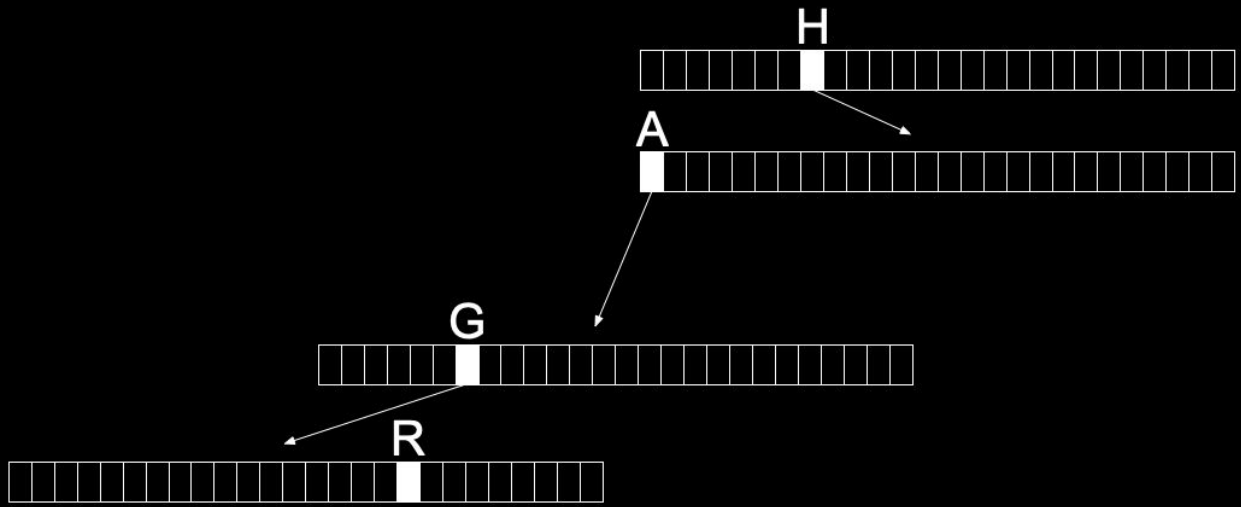
tries

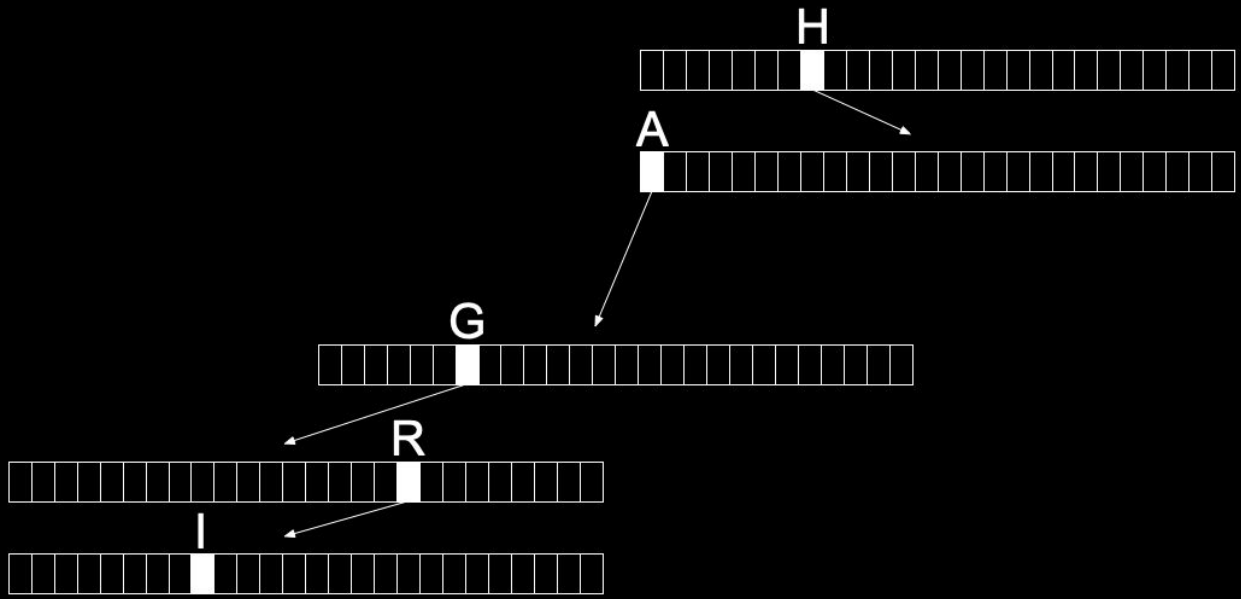

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

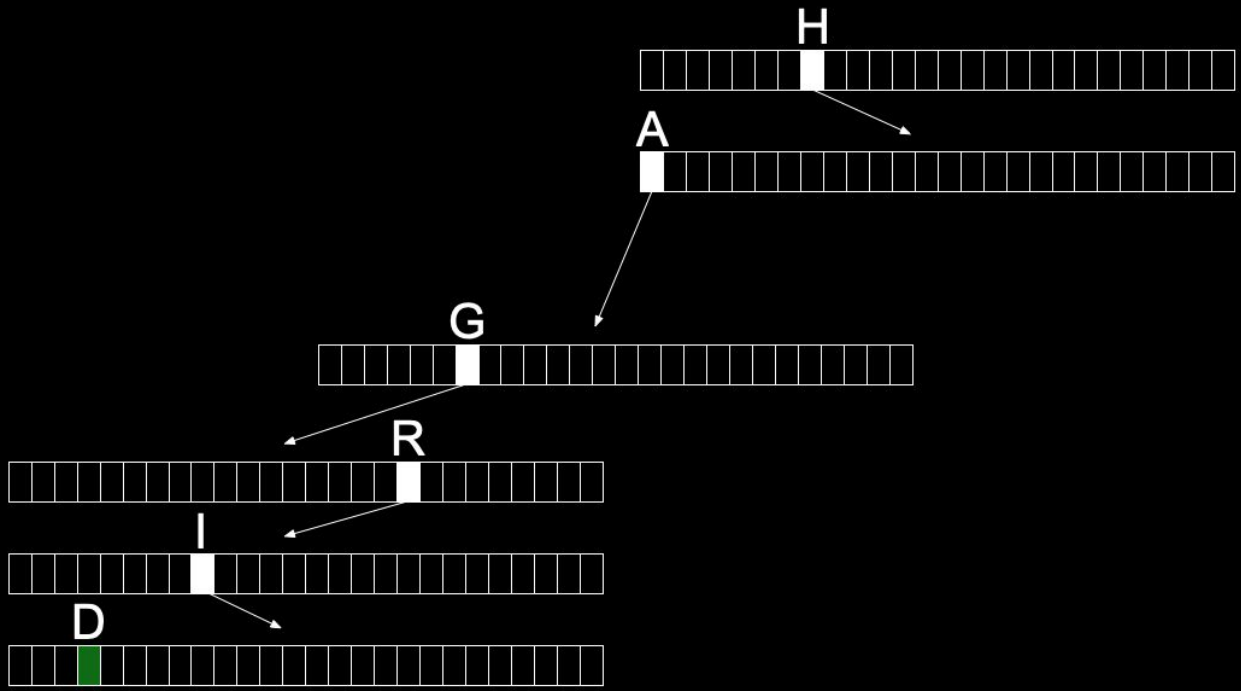

H

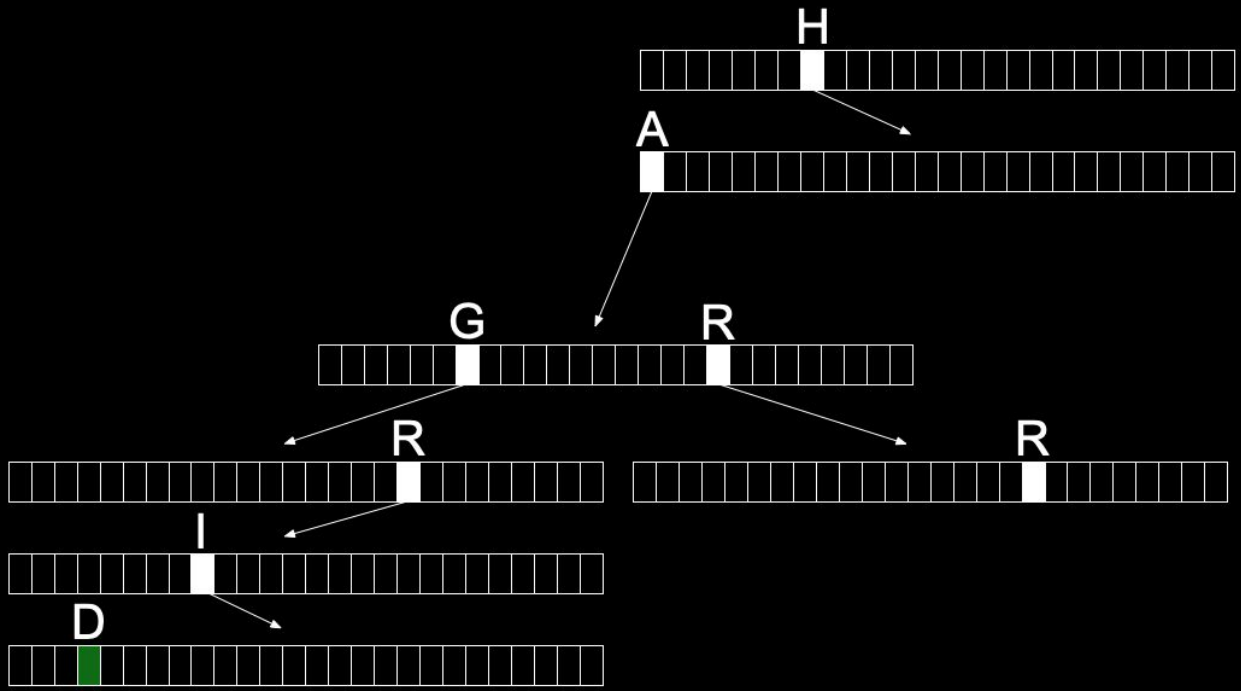

H
A

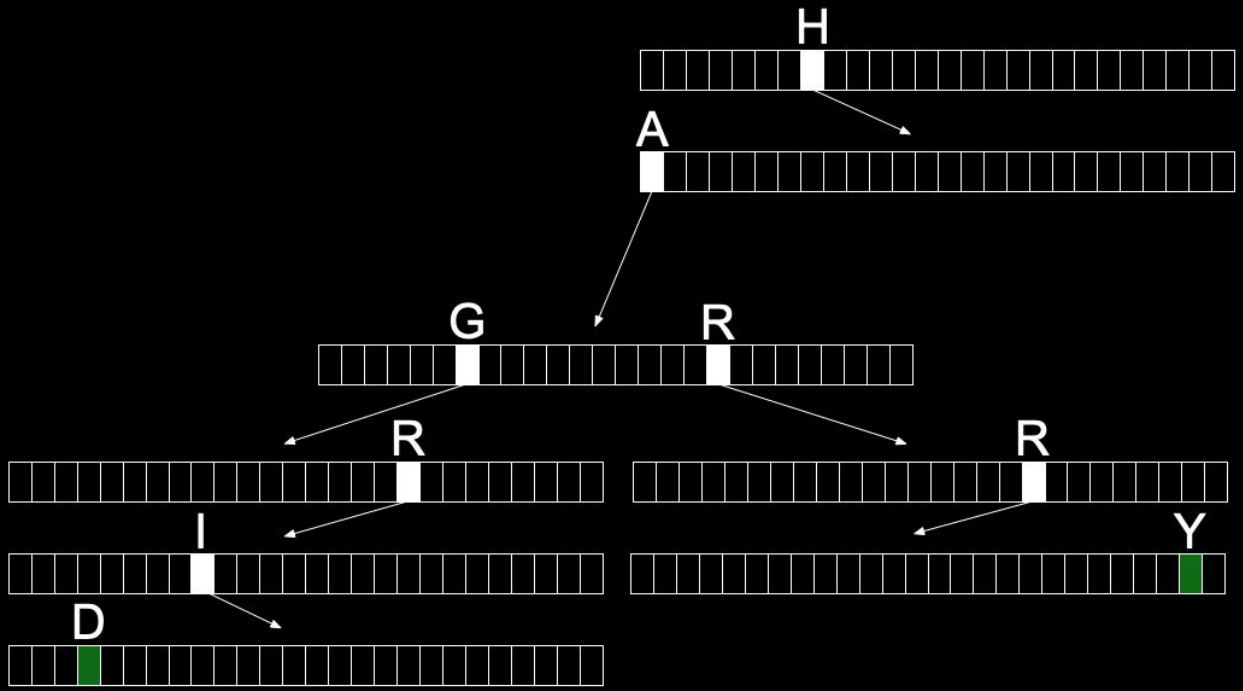

H
A
G

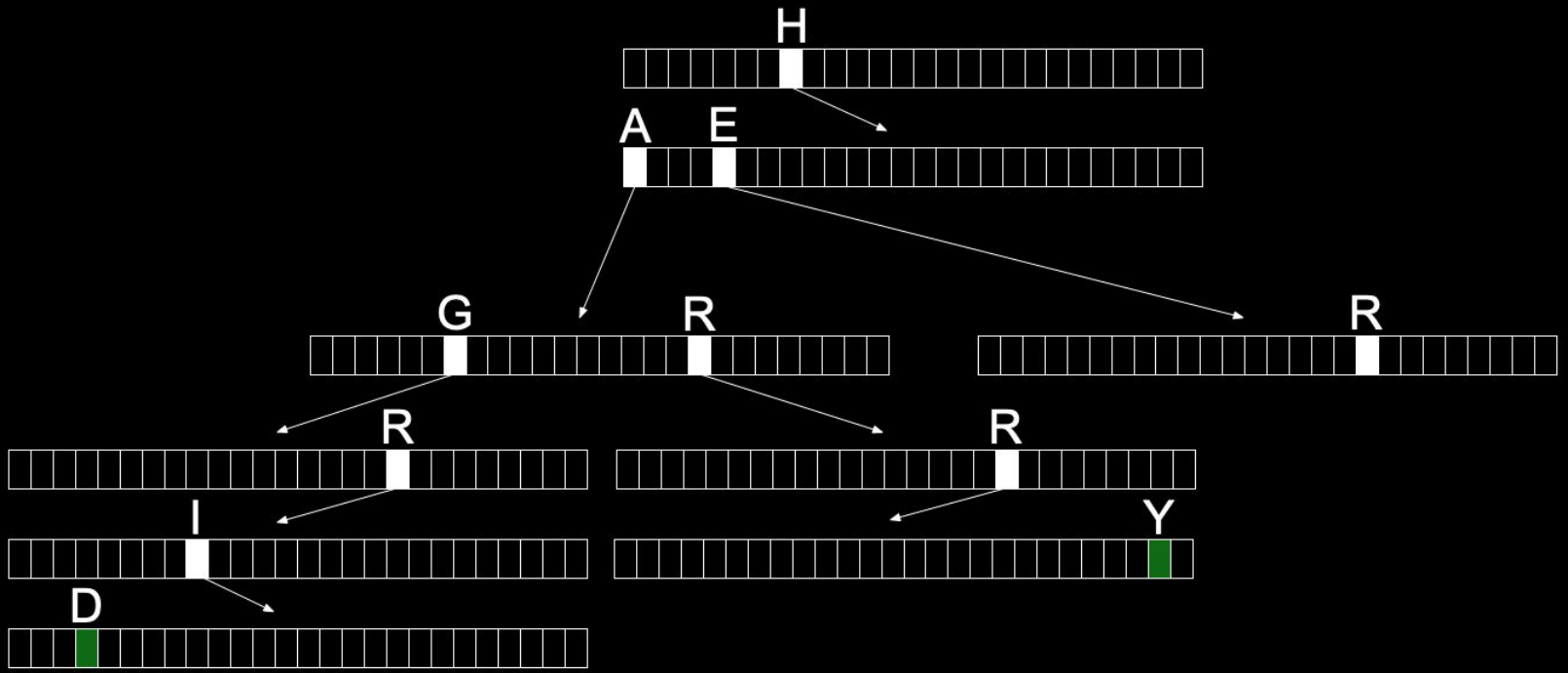

H
A
G
R

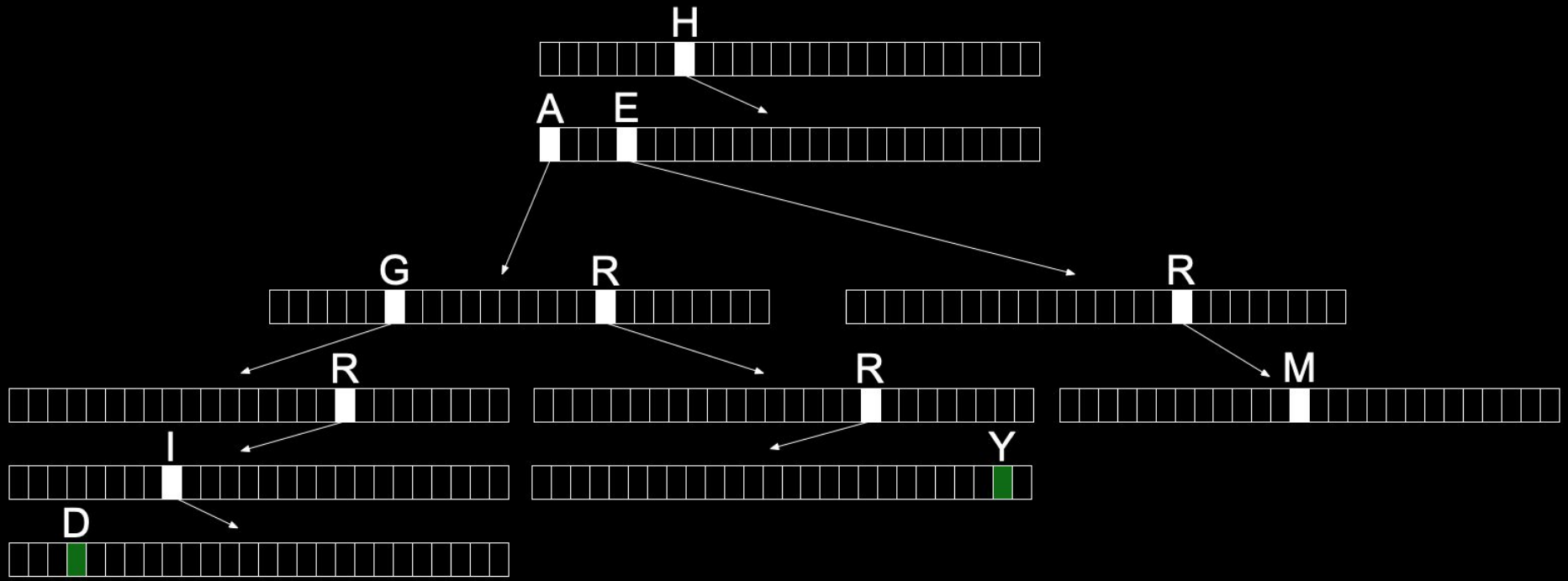

H
A
G
R
I

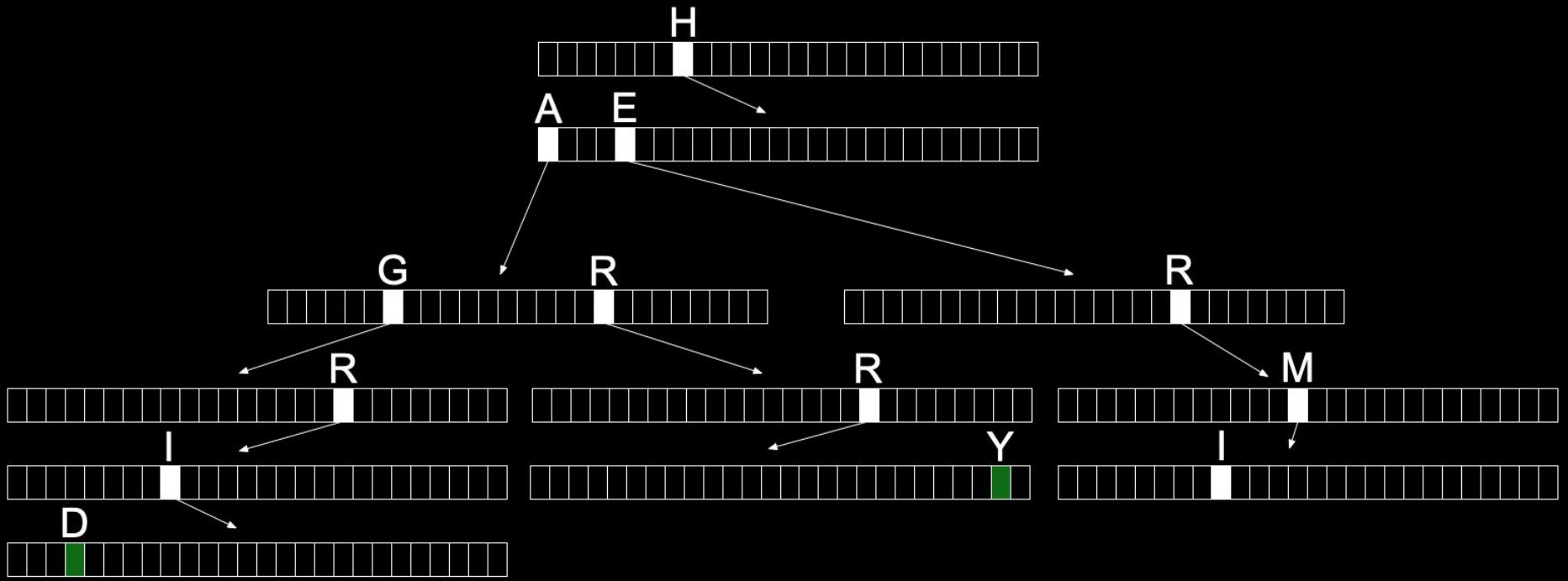

H
A
G
R
I
D

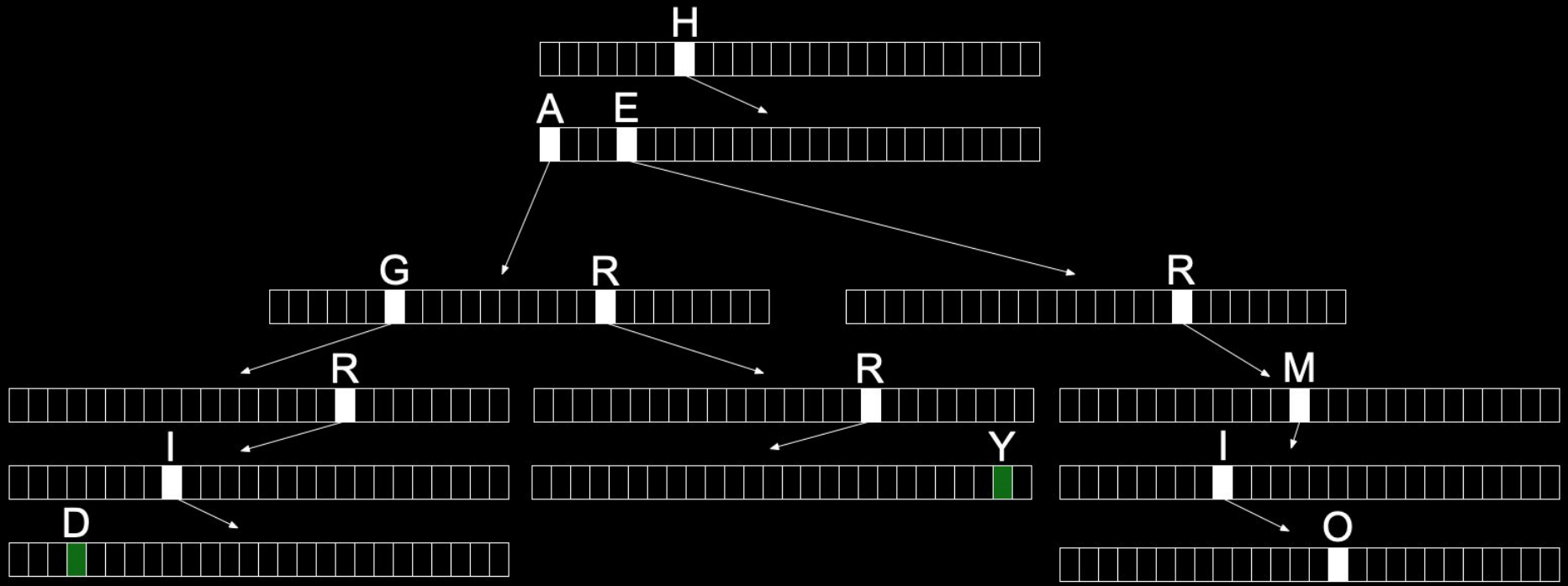

H
A
G
R
R
R
I
D

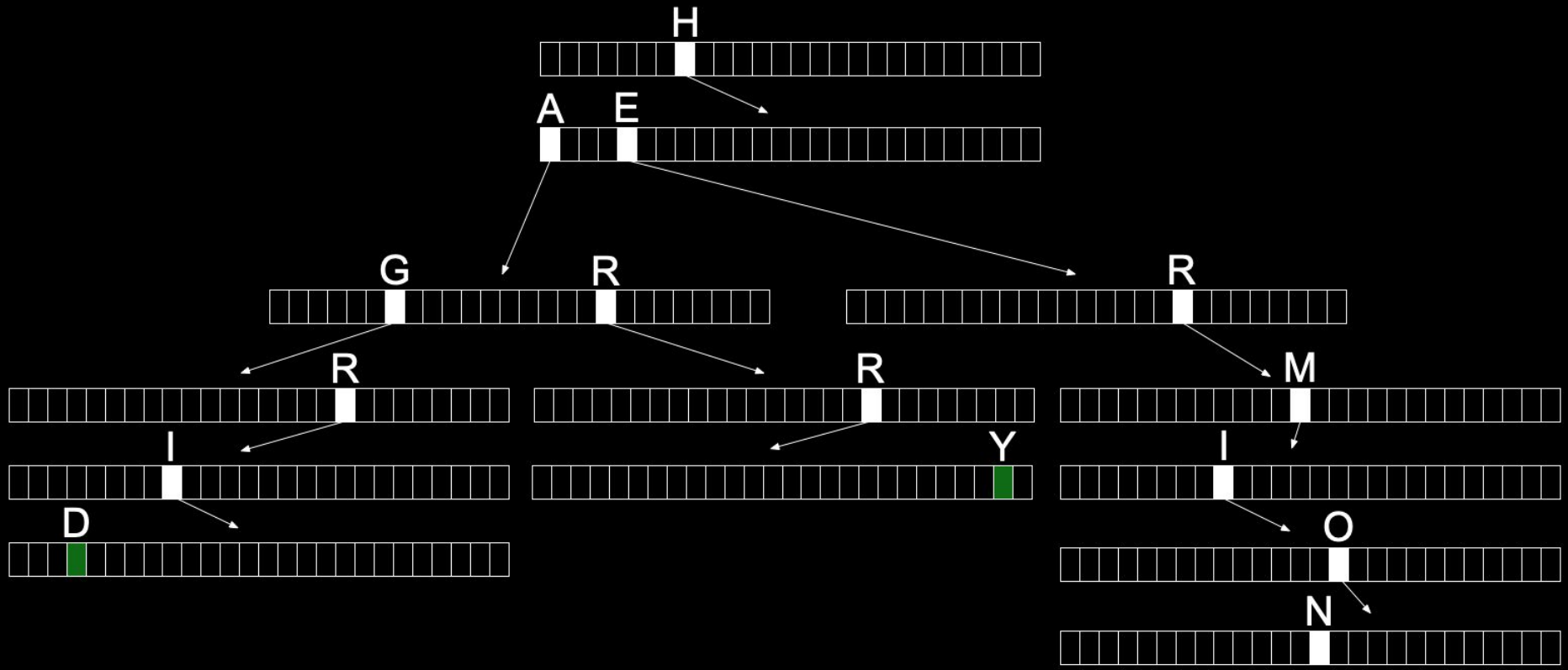

H
A
G
R
R
R
I
Y
D

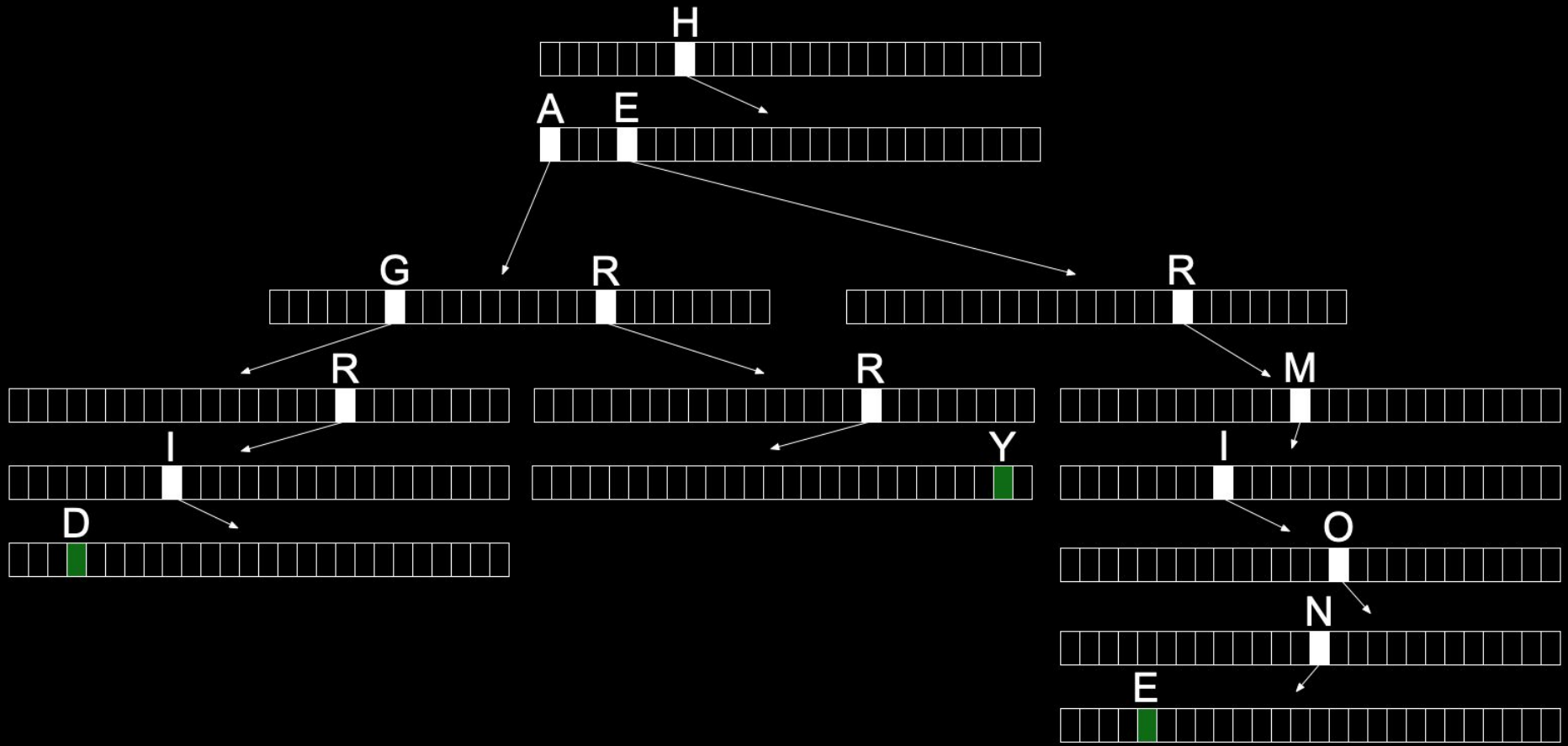

H
A
E
G
R
R
R
R
I
Y
D


H
A
E
G
R
R
R
R
M
I
Y
D


H
A
E
G
R
R
R
R
M
I
Y
I
D


H
A
E
G
R
R
R
R
M
I
Y
I
D
O


H
A
E
G
R
R
R
R
M
I
Y
I
D
O
N


H
A
E
G
R
R
R
R
M
I
Y
I
D
O
N
E

$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$

$O(k)$ search

$O(k)$ search, insert

$O(1)$ search, insert

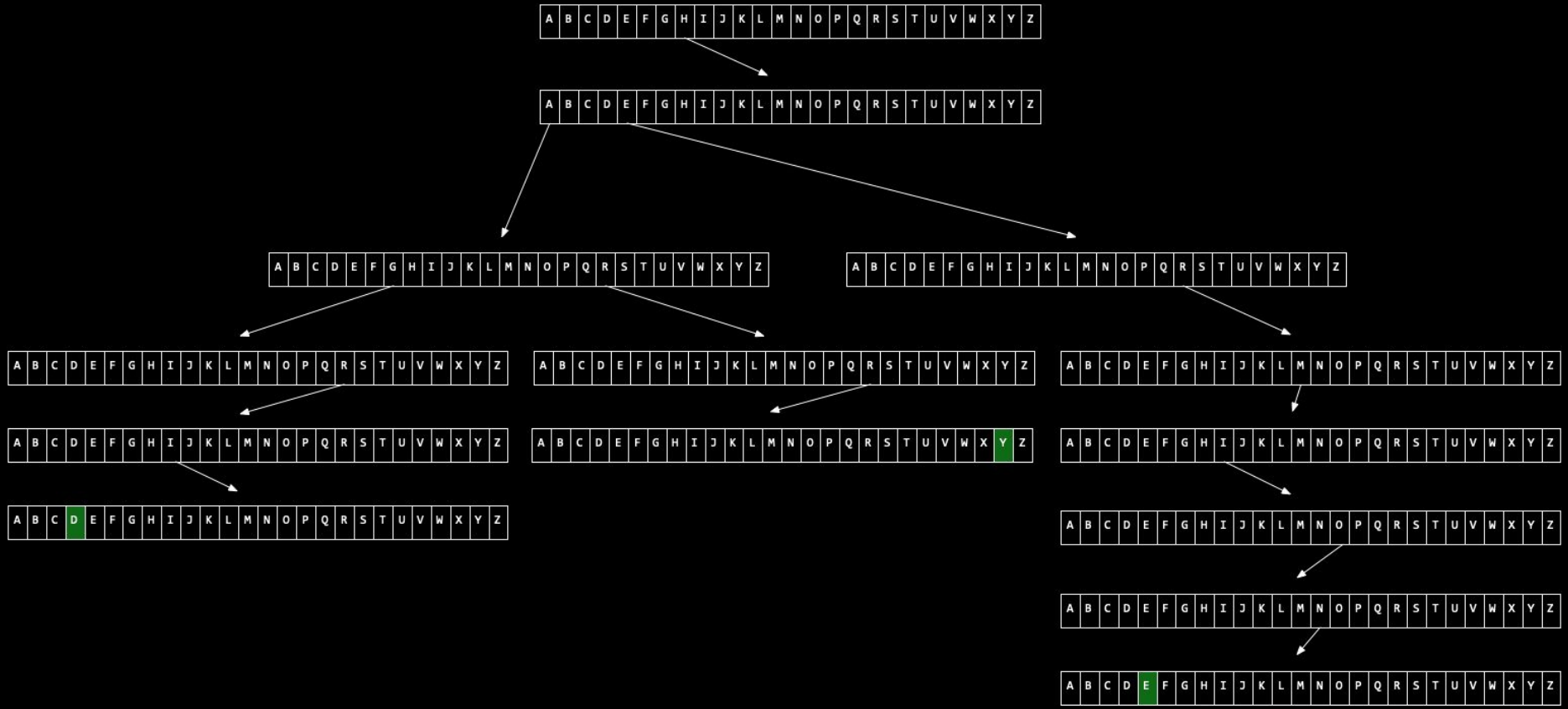
$O(n^2)$

$O(n \log n)$

$O(n)$

$O(\log n)$

$O(1)$ search, insert


A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

abstract data structures

queues

FIFO

enqueue

dequeue

stacks

LIFO

push

pop

dictionaries

This is CS50