CS50 Quiz Preparation

Quiz Logistics

Quiz Logistics

- Released Sat 11/10 noon, due Thu 11/15 noon
- Open-book for all non-human resources
- Submit the quiz via `submit50`

How to Prepare

- Review lecture notes, source code, slides, video
- Attend or watch review session
- Take past years' tests and quizzes
- Review problem set specifications, sample solutions, distribution code

Themes

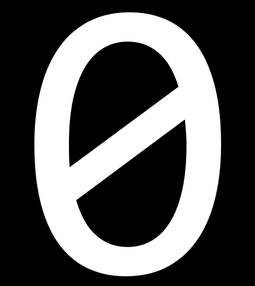
Themes

- Representing Data
- Abstraction
- Algorithms
- Trade-Offs
- Security

Week 0

Computational Thinking, Scratch

Binary



Bit

Binary

1

Bit

Binary

00011100

Byte

Binary

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

00011100

Byte

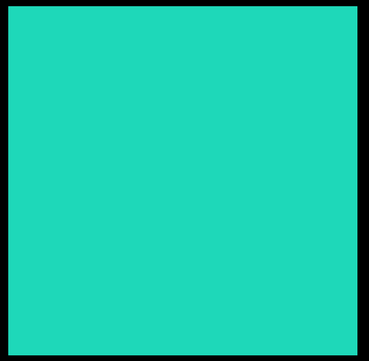
ASCII

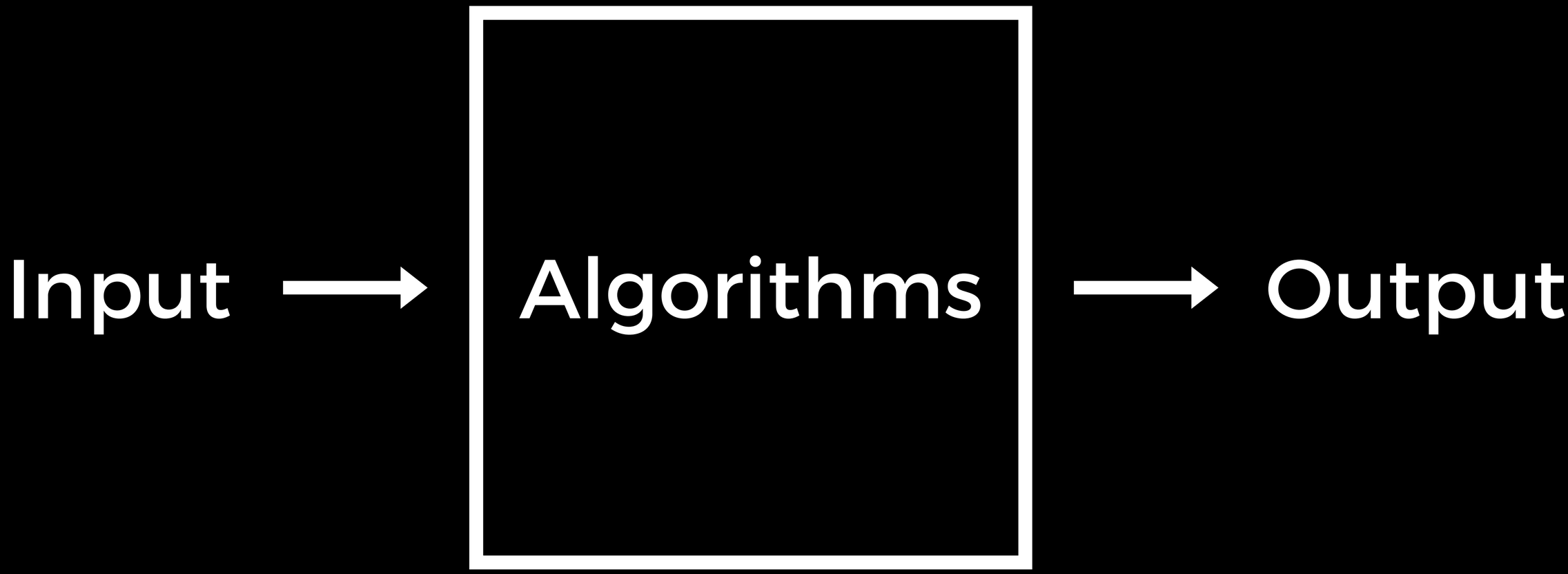
A	B	C	D	E	…	Z
65	66	67	68	69	...	90

a	b	c	d	e	…	z
97	98	99	100	101	...	122

RGB

30 216 185



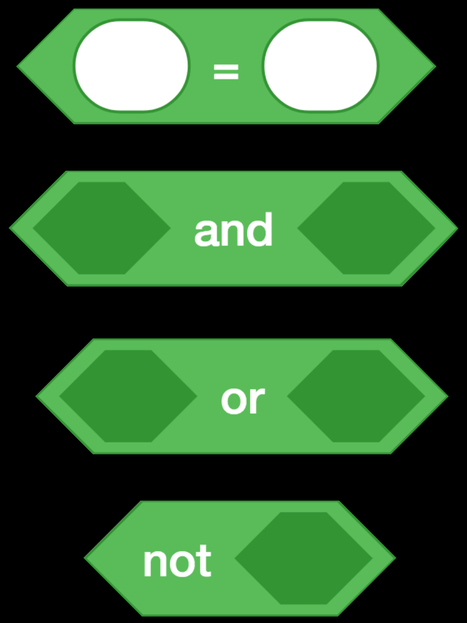

Input
Algorithms
Output

Functions

Variables

Conditions

Boolean Expressions



Loops

Week 1

C

```c
#include <stdio.h>

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

Types

- `int`
- `char`
- `long`
- `float`
- `double`
- `string`
- `bool`
- `...`

Variables

```
string answer = get_string("What's your name?\n");
printf("%s\n", answer);
```

Conditions

```
if (x < y)
{
    printf("x is less than y\n");
}
else if (x > y)
{
    printf("x is greater than y\n");
}
else
{
    printf("x is equal to y\n");
}
```

Loops

```
for (int i = 0; i < 10; i++)
{
    ...
}
```

```
while (x < 10)
{
    ...
}
```

```
do
{
    ...
}
while (x < 10);
```

Functions

```
void cough(int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("cough\n");
    }
}
```

Floating-Point Imprecision

```
x: 2
y: 10
x / y = 0.200000002980232238769531250
```

Integer Overflow

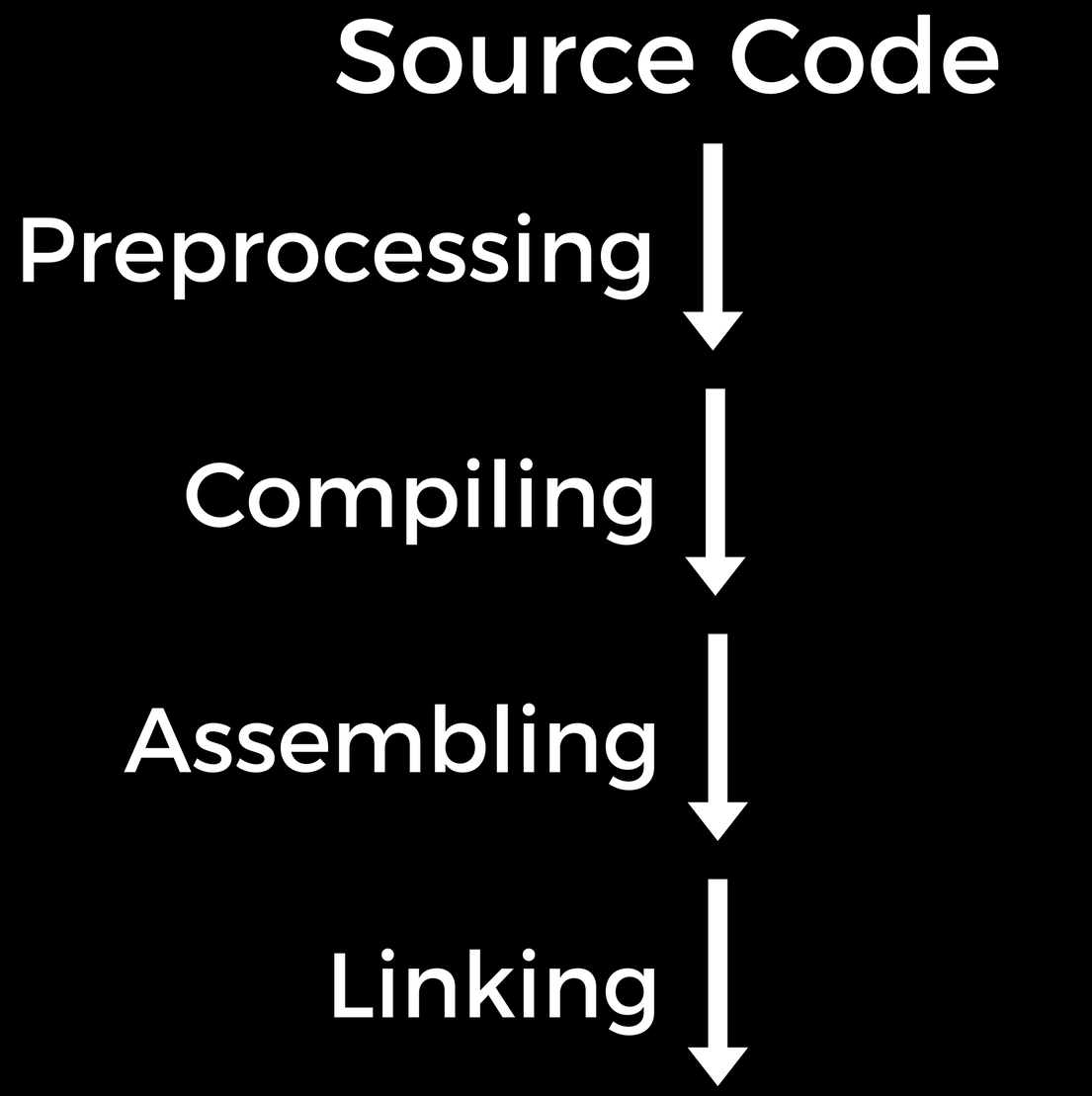
```
#include <stdio.h>
#include <unistd.h>

int main(void)
{
    // Iteratively double i
    for (int i = 1; ; i *= 2)
    {
        printf("%i\n", i);
        sleep(1);
    }
}
```

Week 2

Arrays

Compiling

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


Source Code
Preprocessing
Compiling
Assembling
Linking
Machine Code

Debugging

- `help50`
- `printf`
- `debug50`

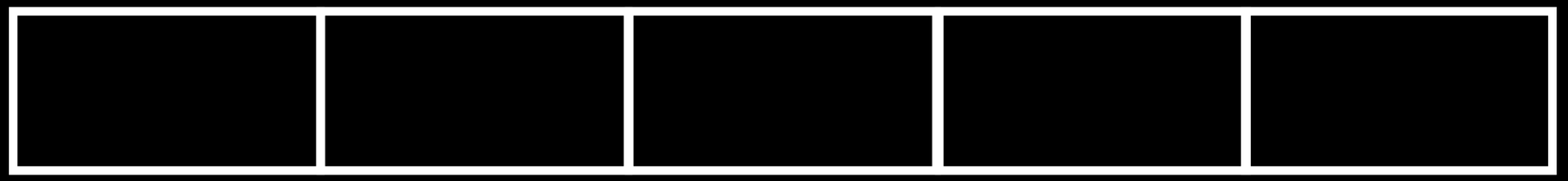
Arrays

value


```
int value = 28;
```

```
int values[5];
```

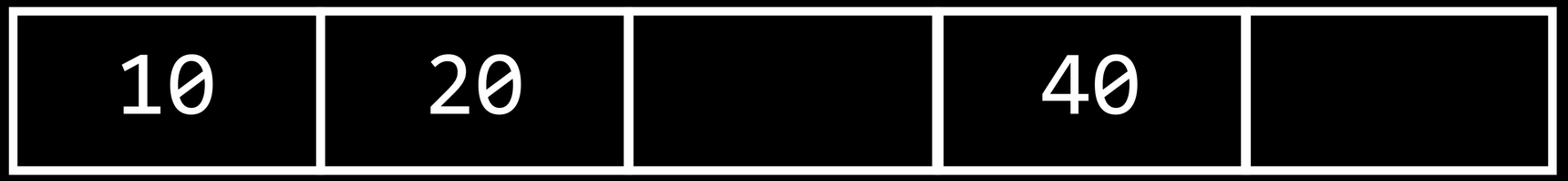
values



Arrays

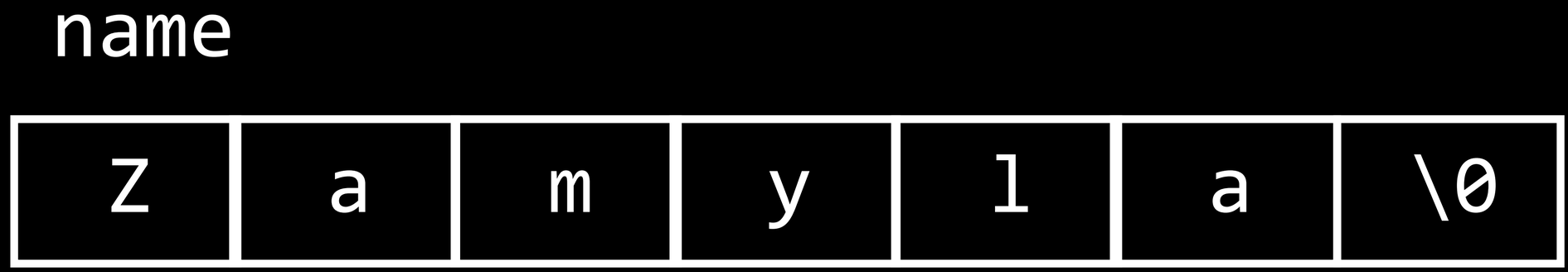
```
int values[5];
values[0] = 10;
values[1] = 20;
values[3] = 40;
```

values



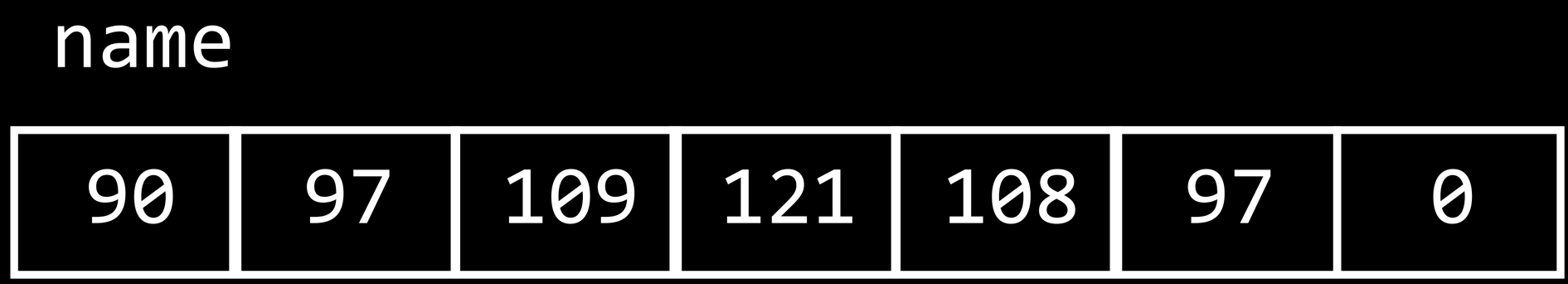
Strings

```
string name = "Zamyla";
```



Strings

```
string name = "Zamyla";
```



Command-Line Arguments

```
$ ./cash

$ make mario

$ clang -o hello hello.c
```

Command-Line Arguments

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

Sorting

- Bubble Sort
- Selection Sort
- Merge Sort

Time Complexity

Algorithm	Time Complexity
Bubble Sort	$O(n^2)$
Selection Sort	$O(n^2)$
Merge Sort	$O(n \log n)$

Week 3

Memory

Pointers

Types

`int`	Integer
`char`	Character

Types

`int *` Address of an Integer

`char *` Address of a Character

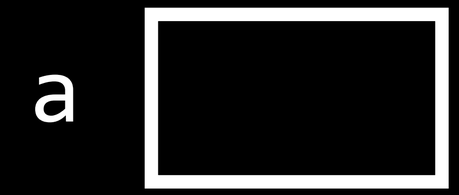
Pointers

- `type *` declares a pointer that stores the address of a `type`
- `*x` takes pointer `x` and gets the value at the address
  - "dereference" operator
- `&x` takes variable `x` and gets its address

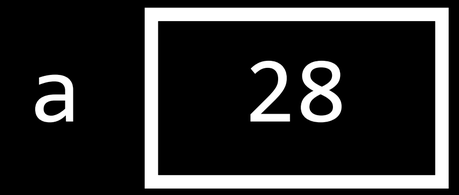
```
int a = 28;
```

```
int a = 28;
```

```
int a = 28;
```

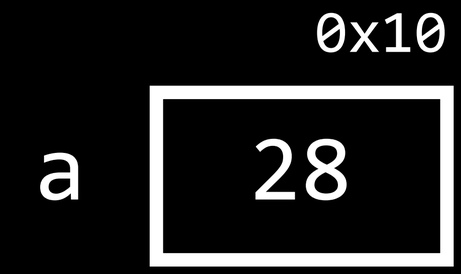


```
int a = 28;
```

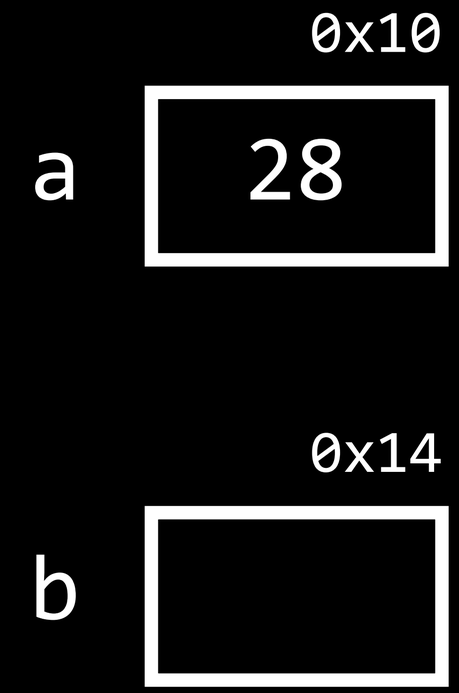


```
int a = 28;
```

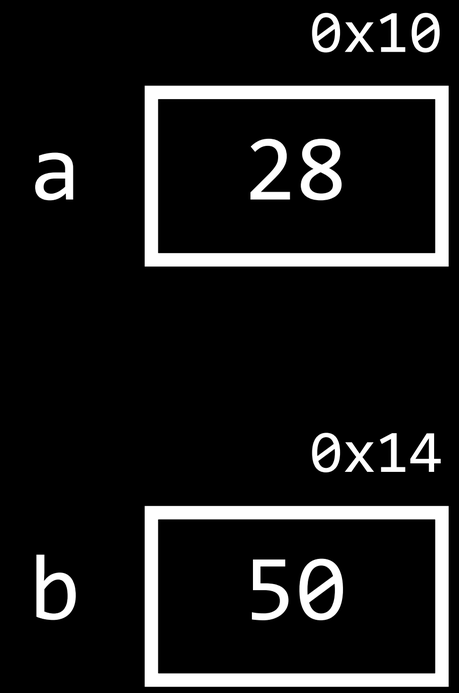
```
int a = 28;

int b = 50;
```



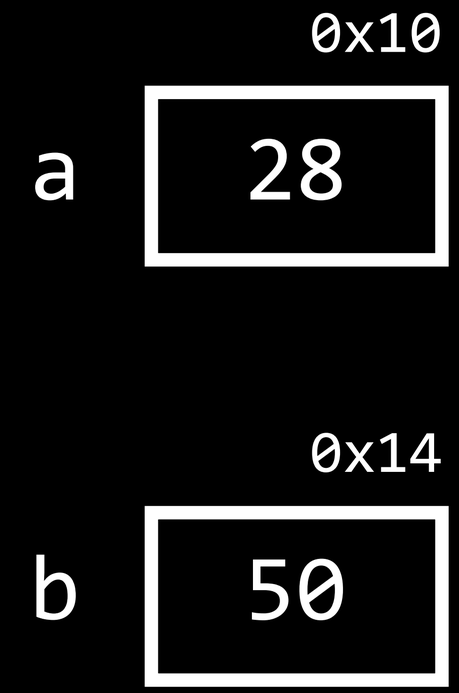
```
int a = 28;

int b = 50;
```



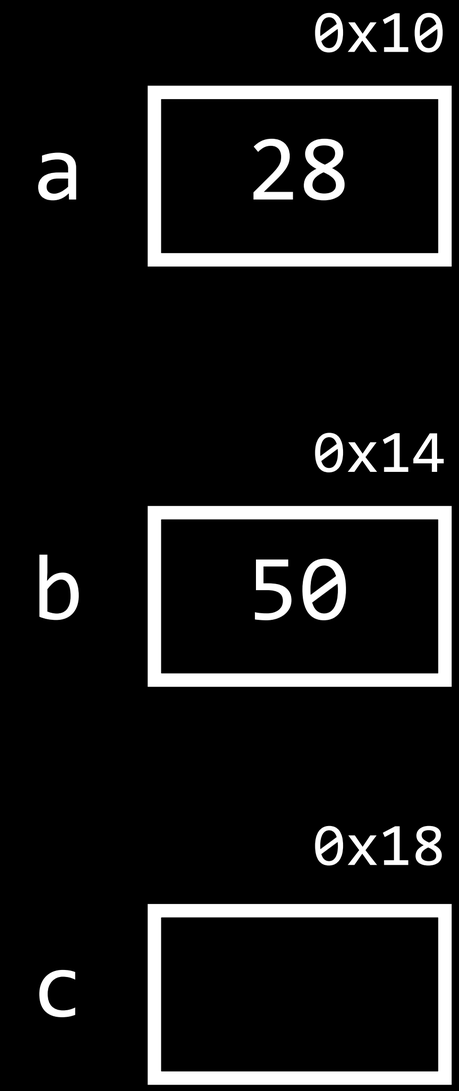
```
int a = 28;

int b = 50;
```



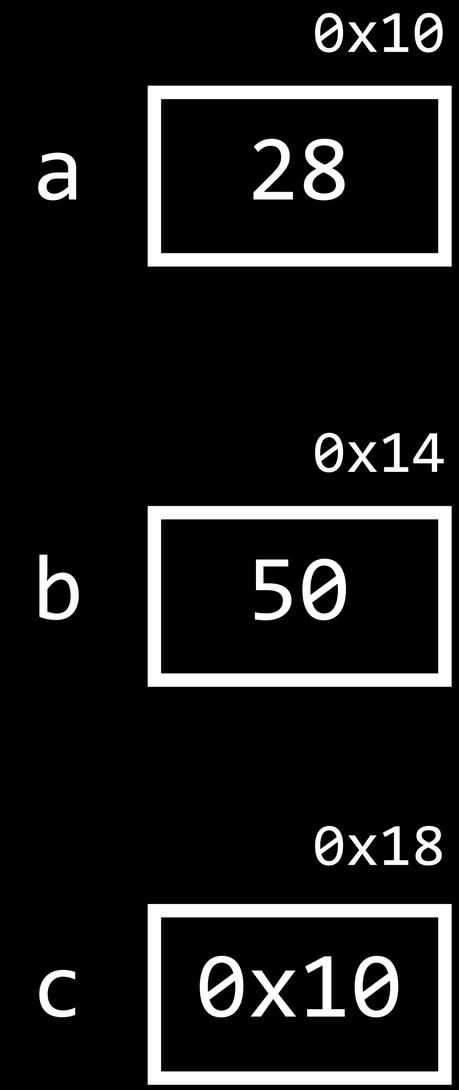
```
int a = 28;

int b = 50;

int *c = &a;
```



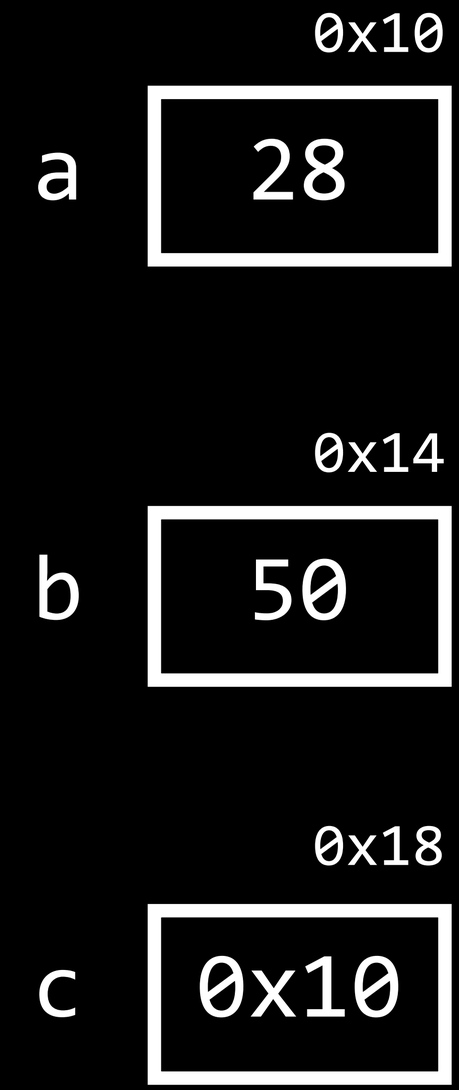
```
int a = 28;

int b = 50;

int *c = &a;
```



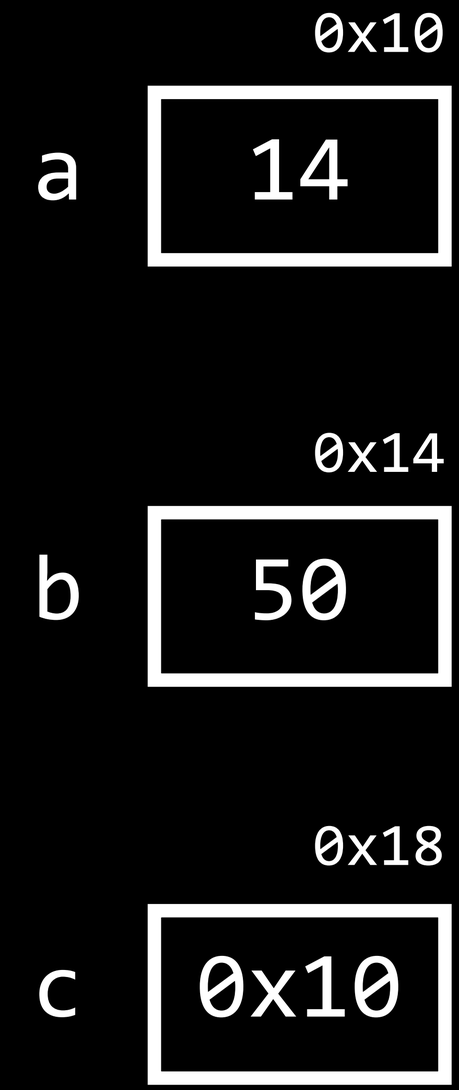
```
int a = 28;

int b = 50;

int *c = &a;
```



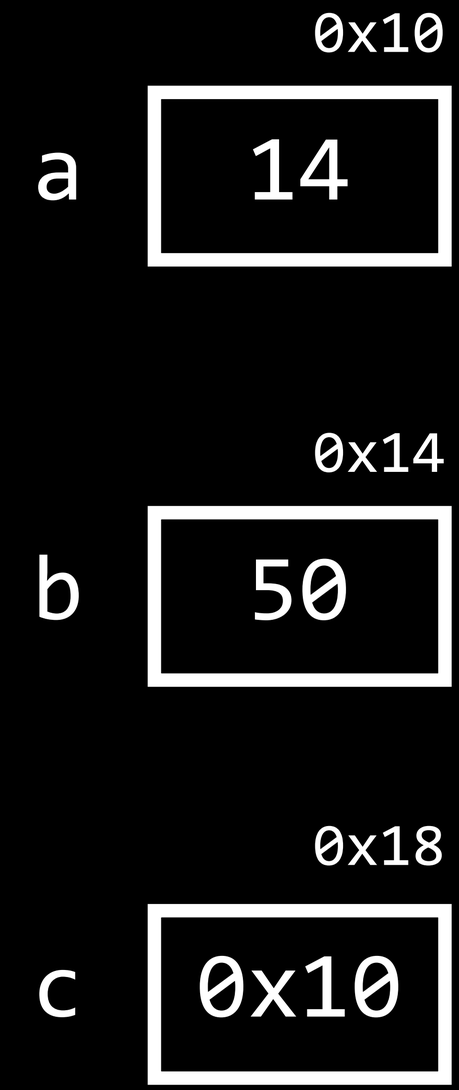
```
int a = 28;

int b = 50;

int *c = &a;

*c = 14;
```



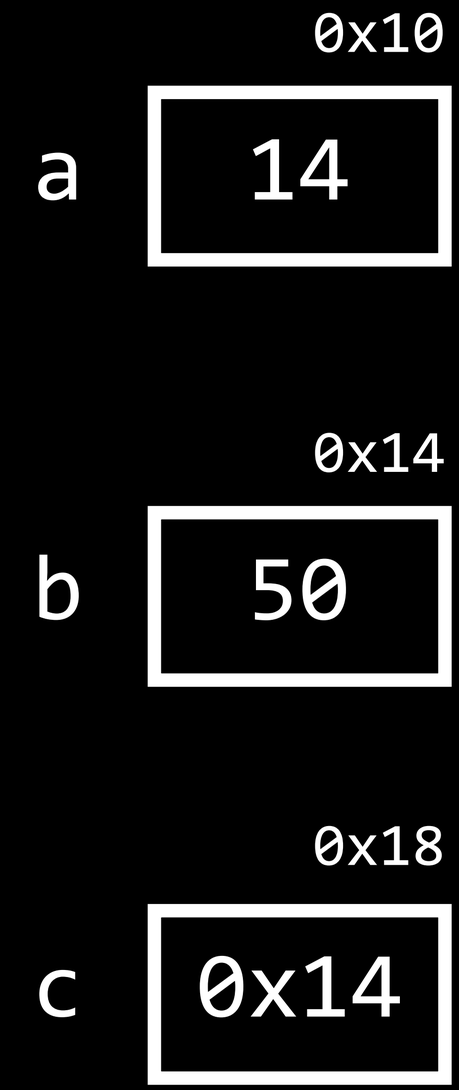
```
int a = 28;

int b = 50;

int *c = &a;

*c = 14;
```



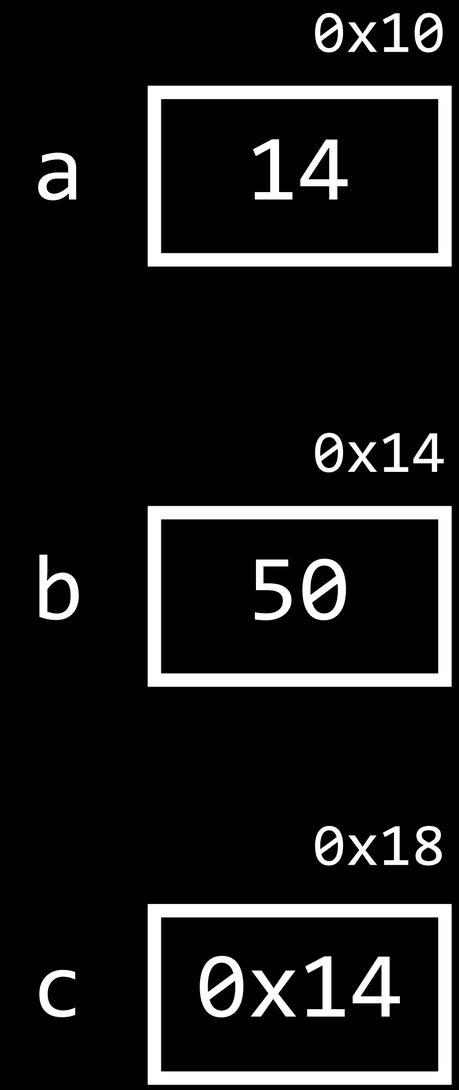
```
int a = 28;

int b = 50;

int *c = &a;

*c = 14;

c = &b;
```



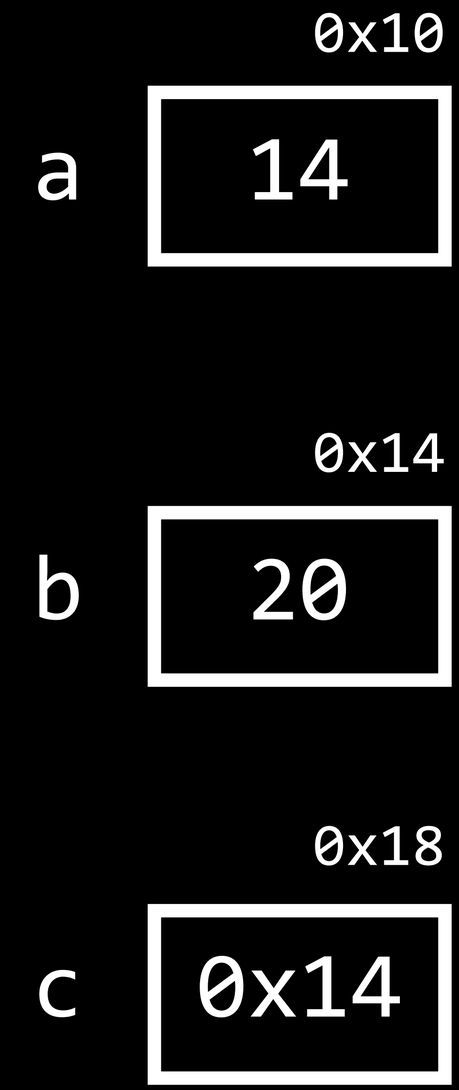
```
int a = 28;

int b = 50;

int *c = &a;

*c = 14;

c = &b;
```



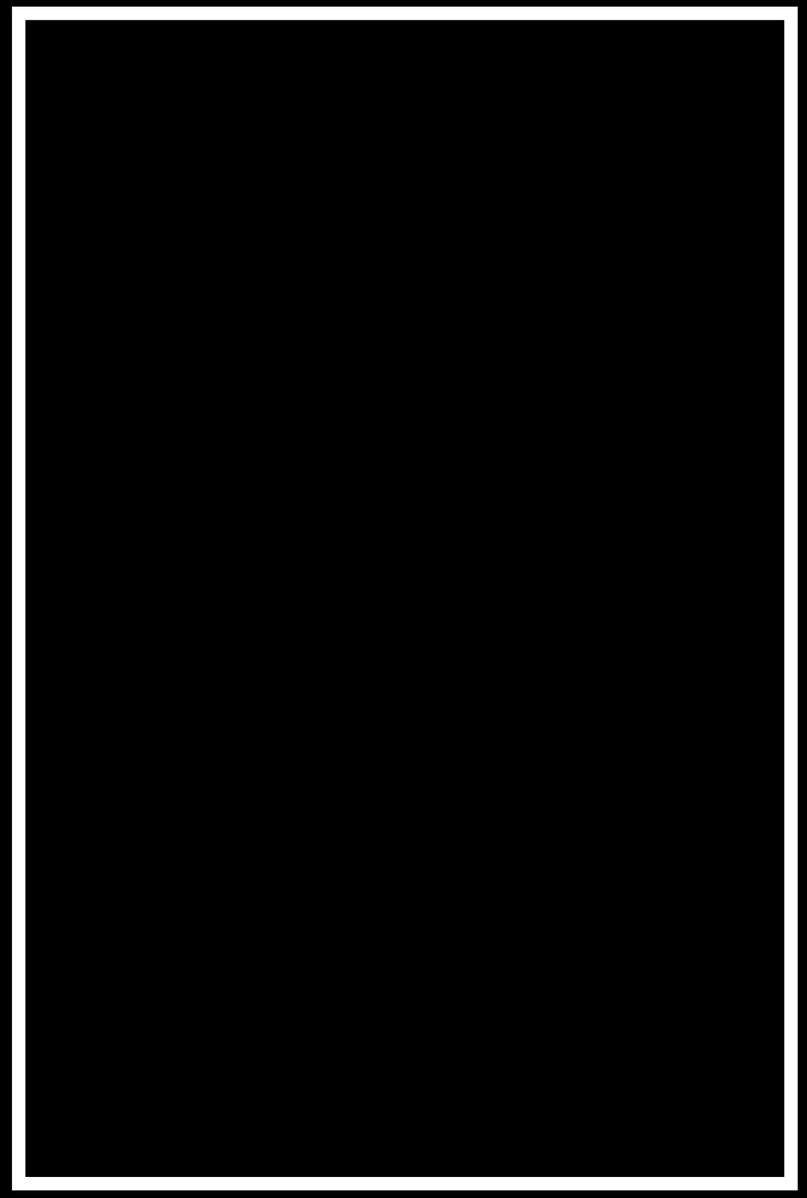
```
int a = 28;

int b = 50;

int *c = &a;

*c = 14;

c = &b;

*c = 20;
```

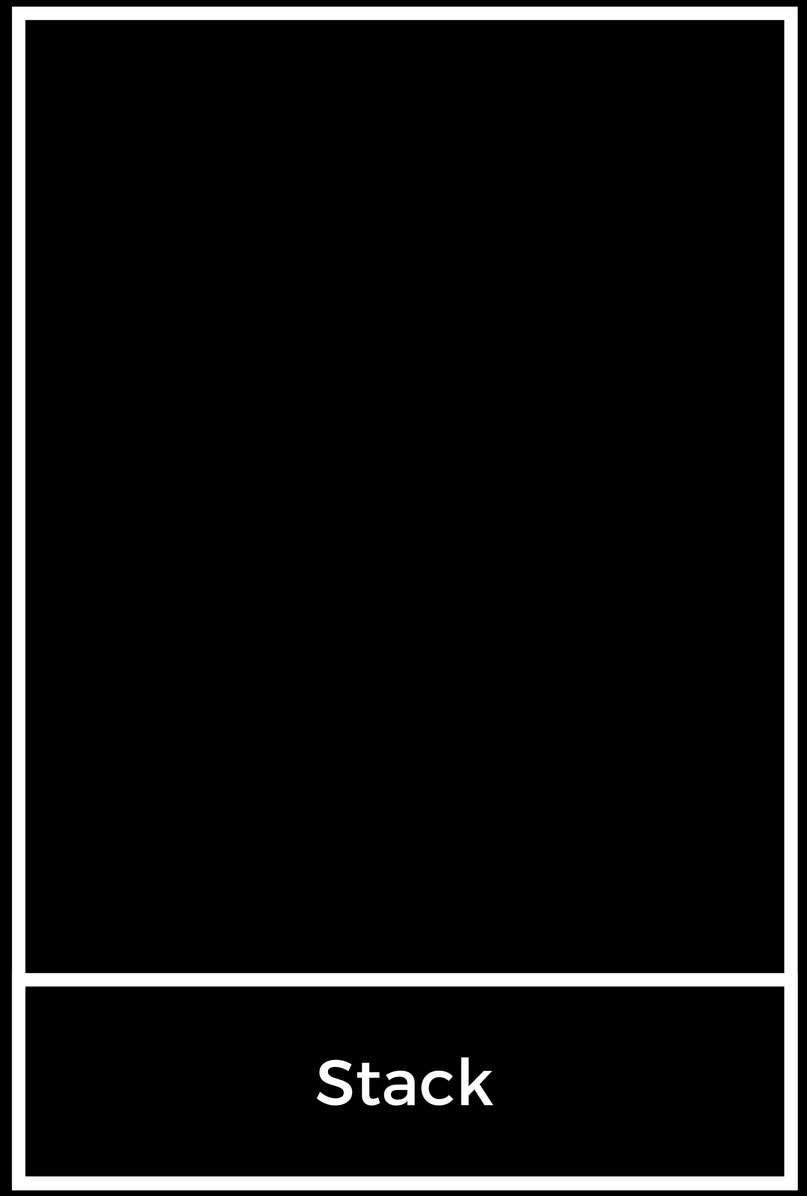


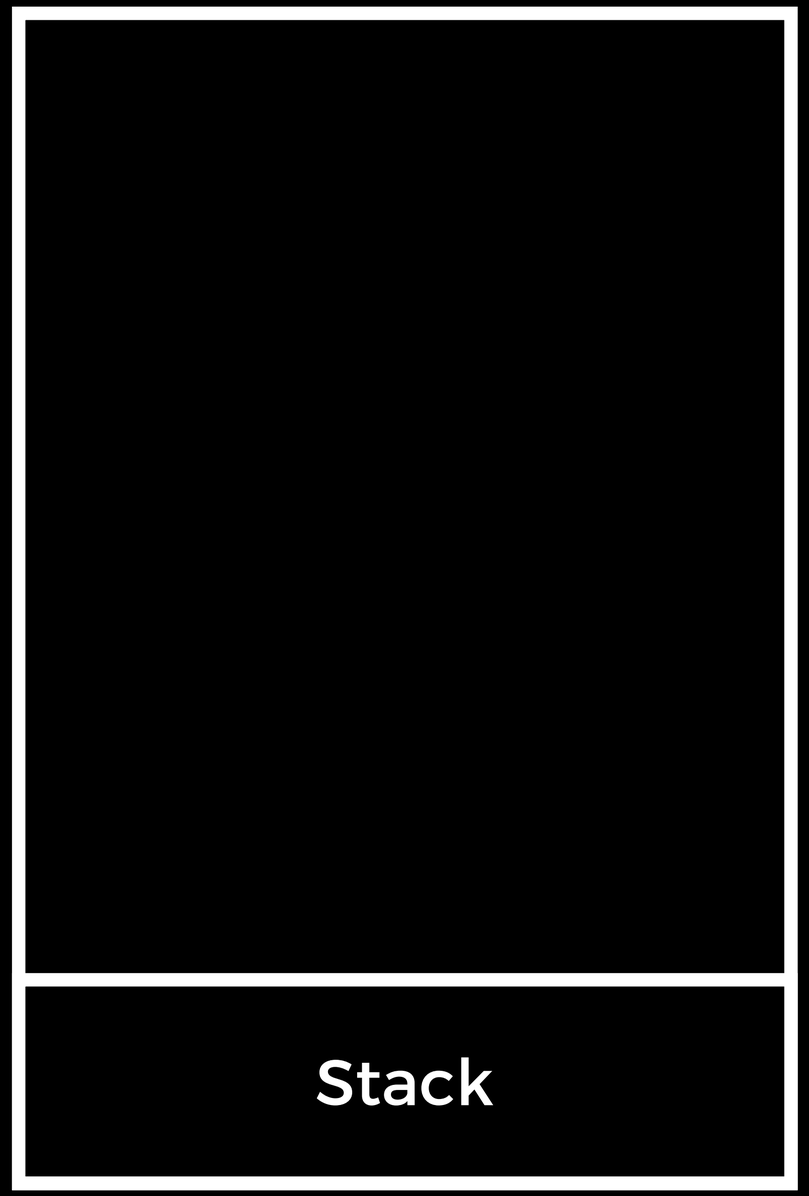
```
int a = 28;

int b = 50;

int *c = &a;

*c = 14;

c = &b;

*c = 20;
```



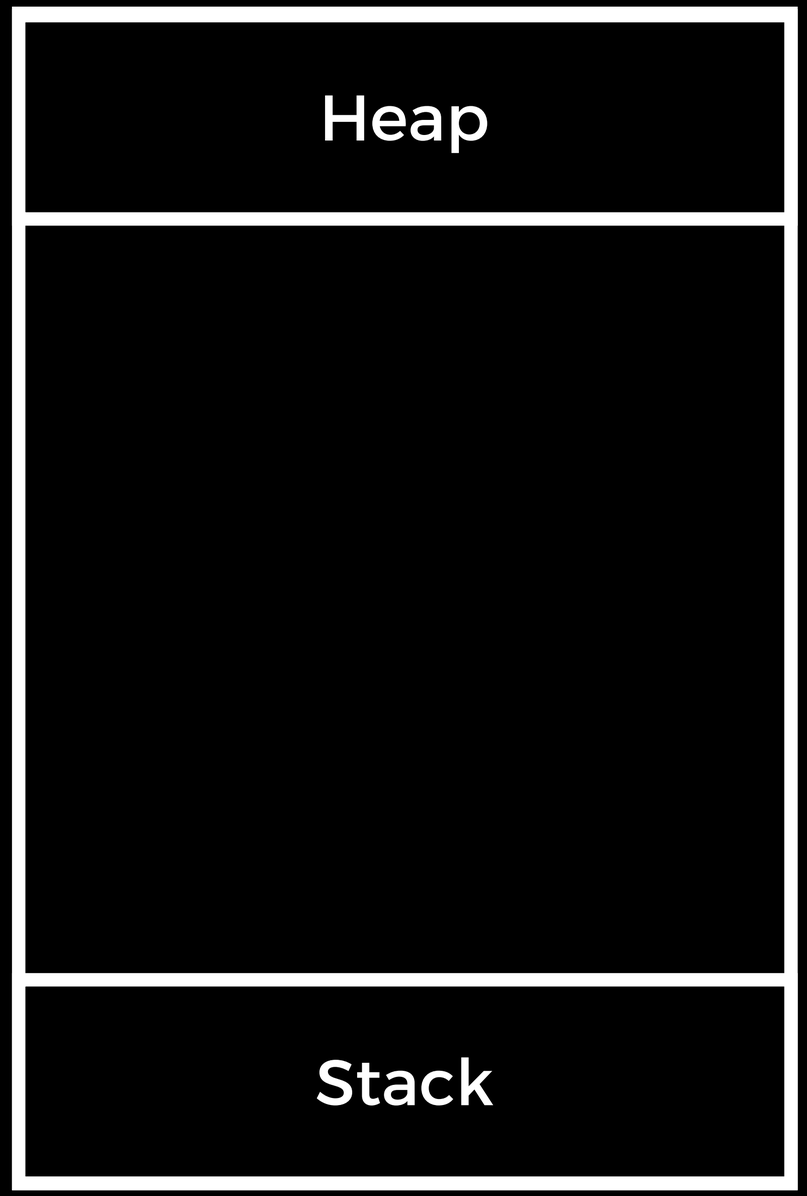
Memory Layout



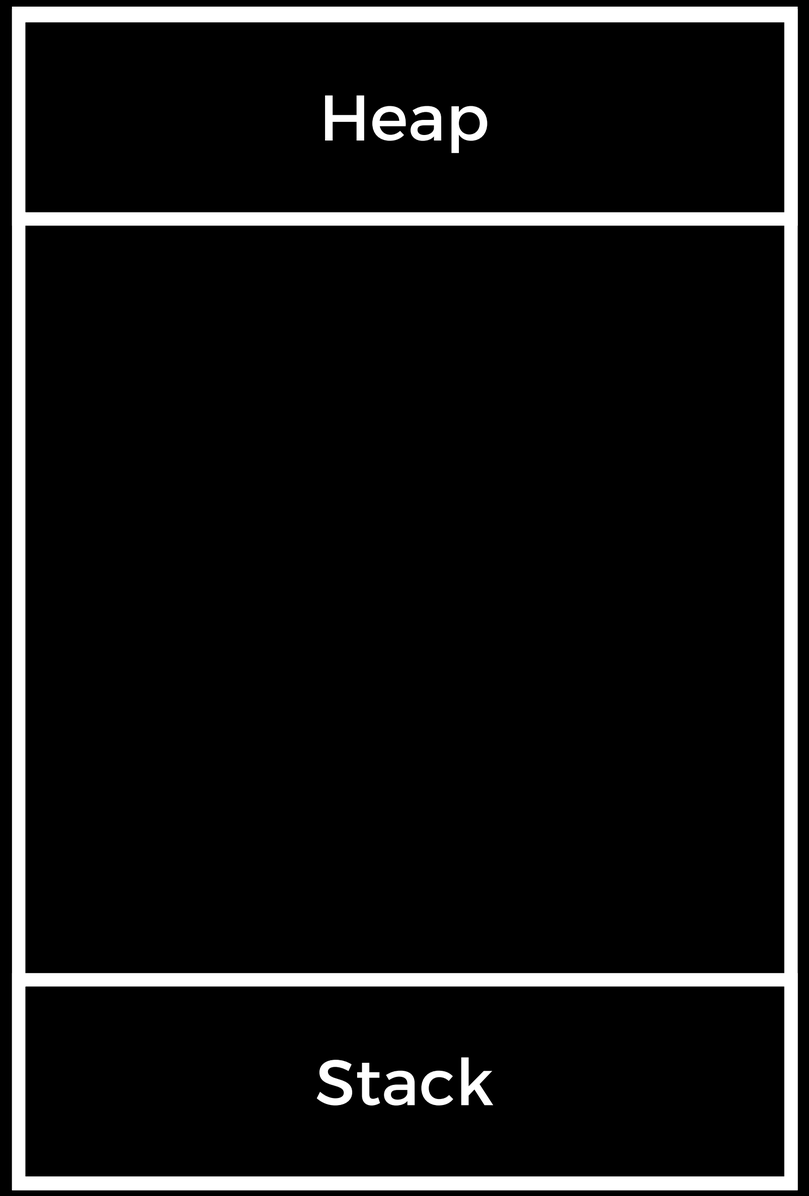

Stack



functions' variables and arguments

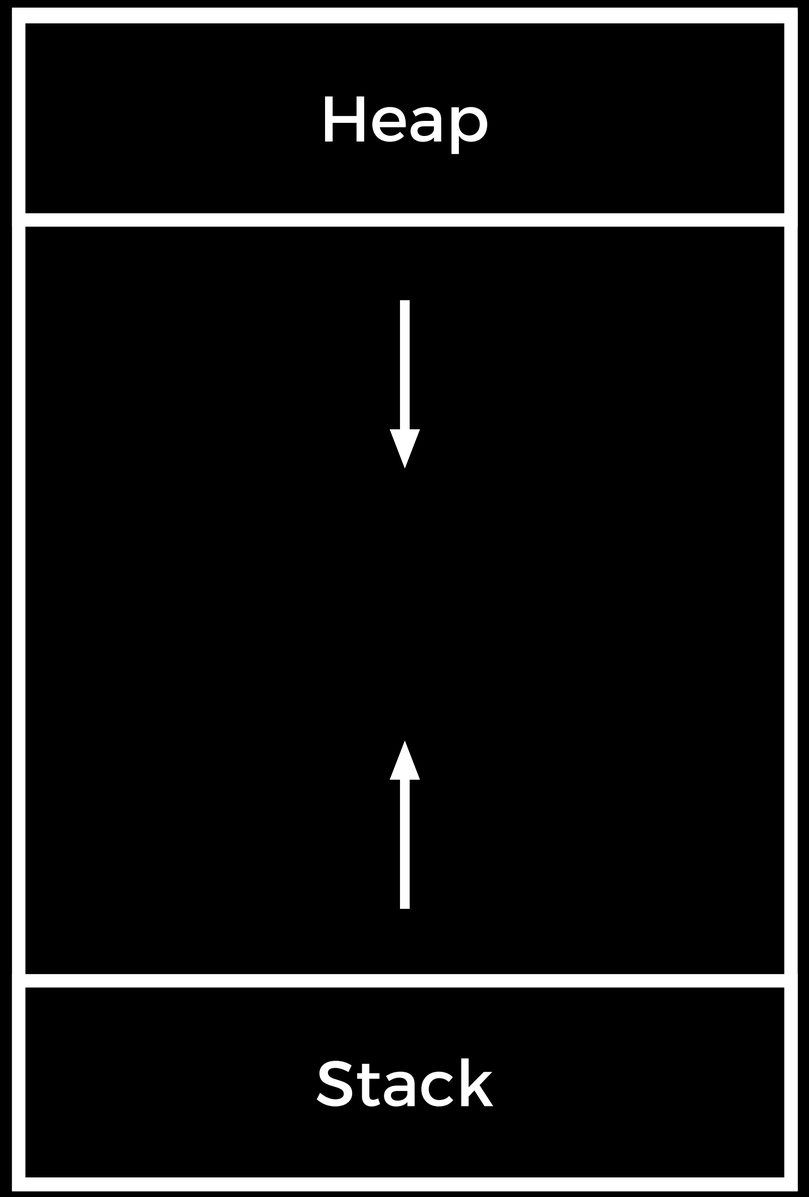


functions' variables and arguments



dynamically allocated memory

functions' variables and arguments



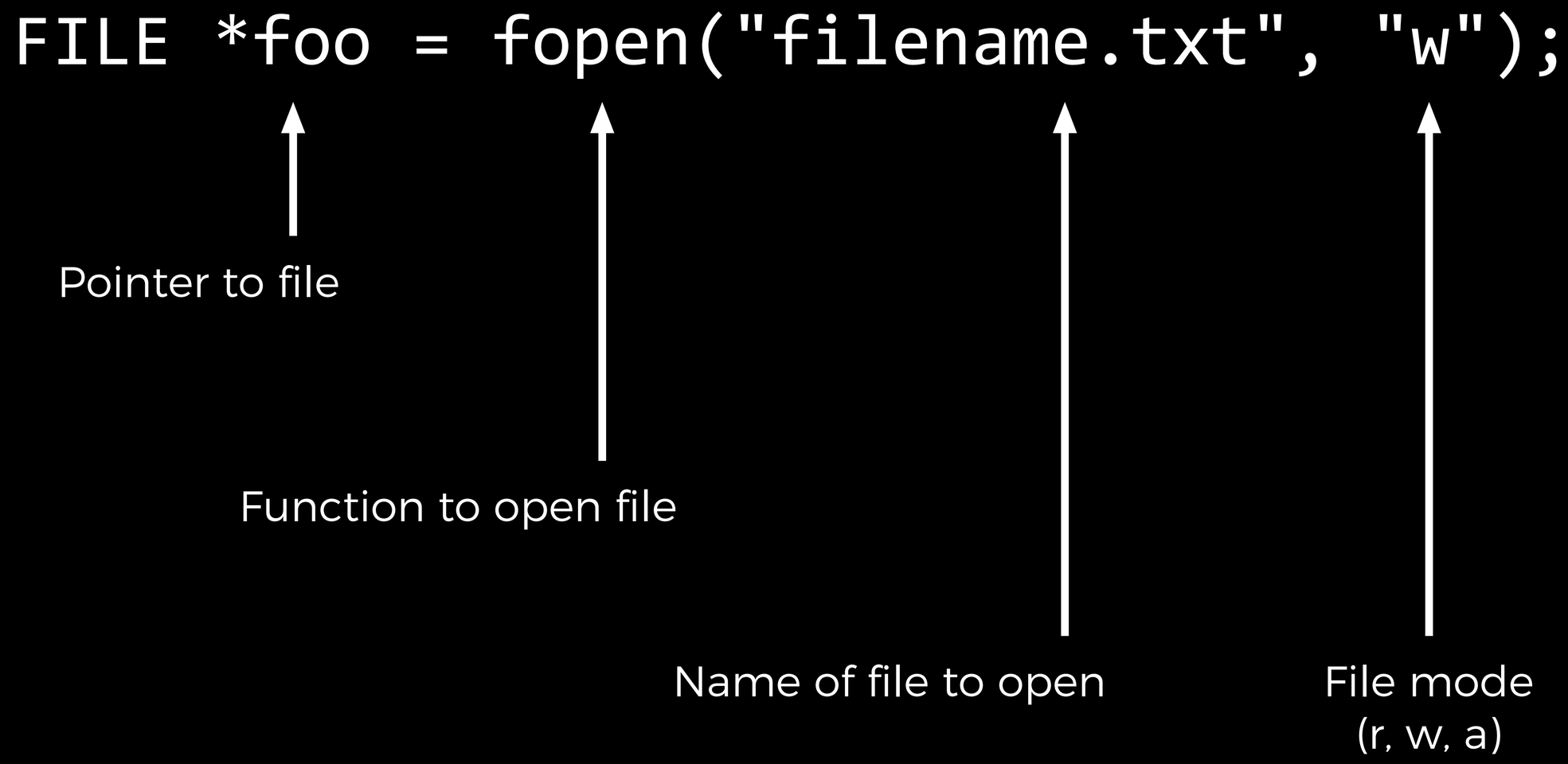
dynamically allocated memory

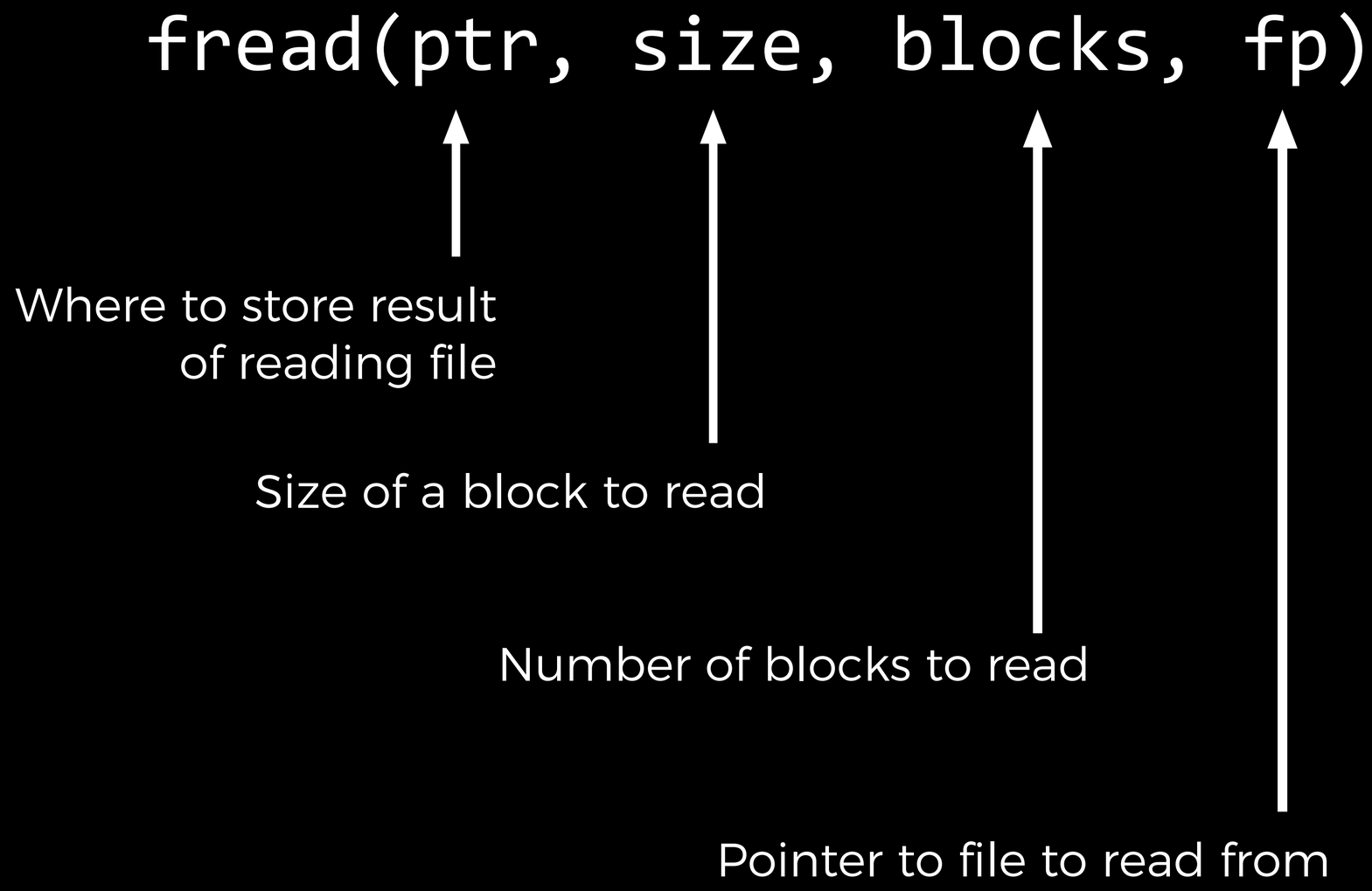
functions' variables and arguments

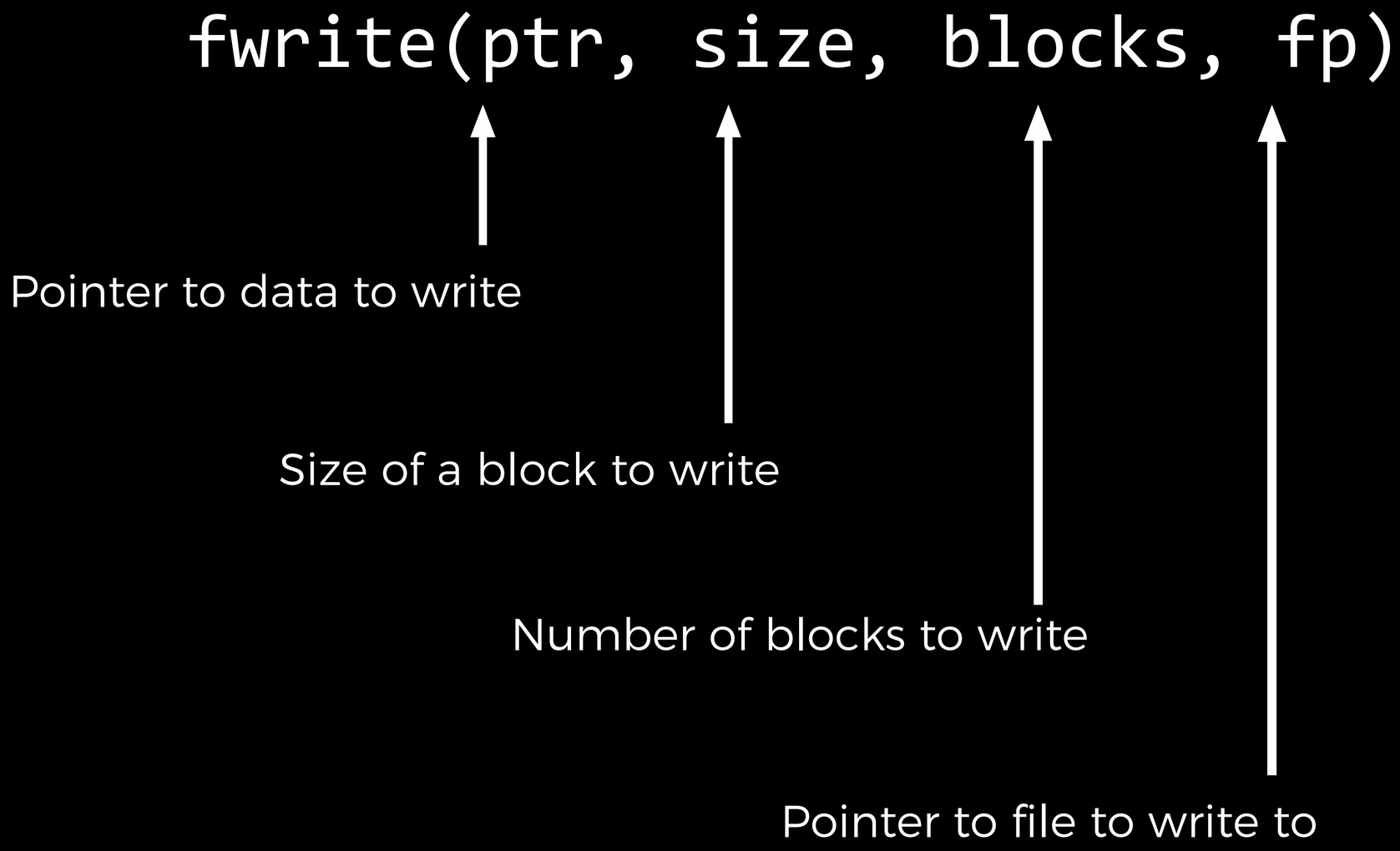
File I/O

File I/O

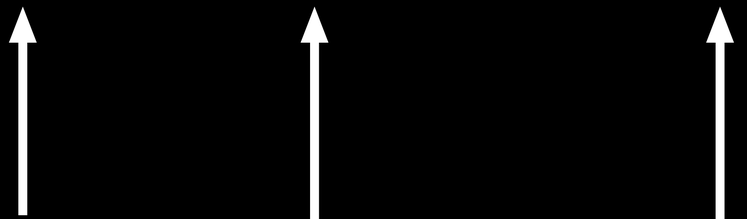
- Opening a file
- Reading from a file
- Writing to a file
- Changing position in a file
- Closing a file


FILE *foo = fopen("filename.txt", "w");
Pointer to file
Function to open file
Name of file to open
File mode
(r, w, a)


fread(ptr, size, blocks, fp)
Where to store result
of reading file
Size of a block to read
Number of blocks to read
Pointer to file to read from


fwrite(ptr, size, blocks, fp)
Pointer to data to write
Size of a block to write
Number of blocks to write
Pointer to file to write to

```
fseek(fp, offset, whence)
```



Pointer to file

How much to move file position by

```
SEEK_SET,
SEEK_CUR,
 SEEK_END
```

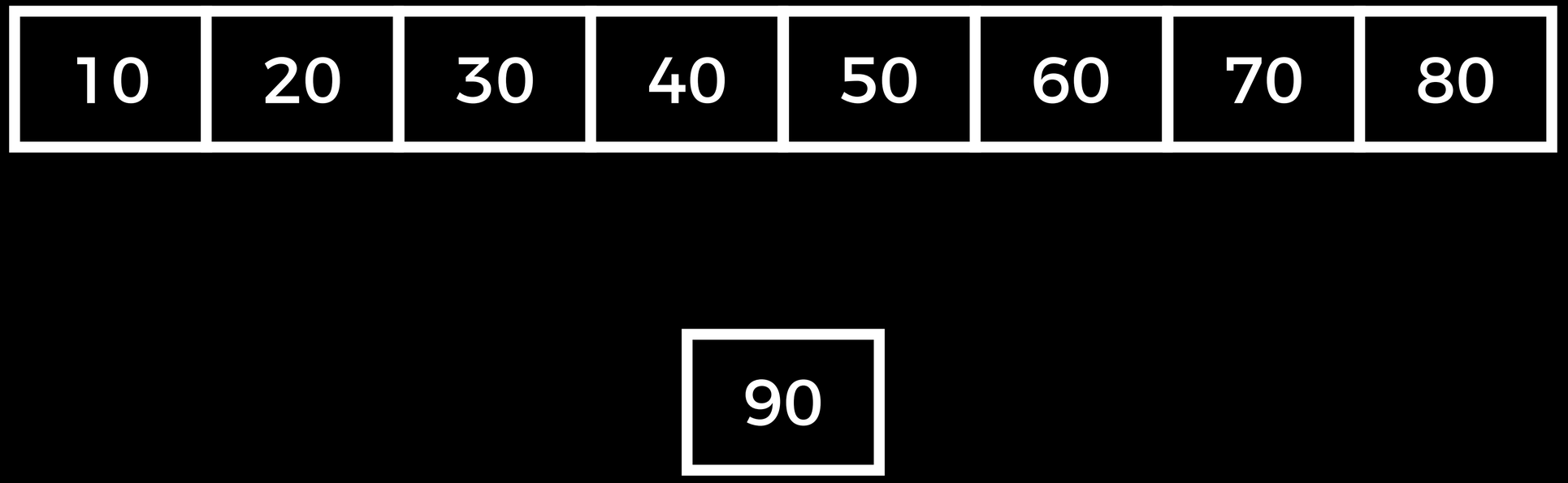
```
fclose(fp)
```

Pointer to file

Week 4

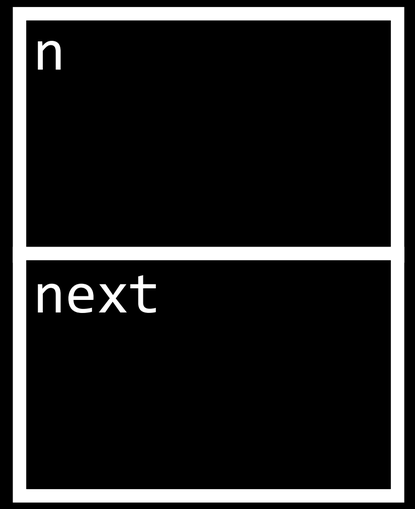
Data Structures

Arrays



Linked List

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



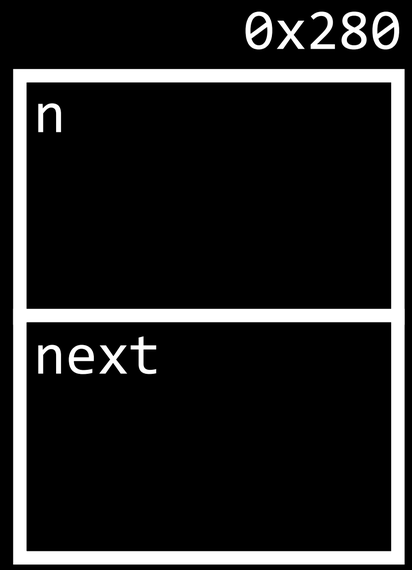
Linked List

Linked List

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

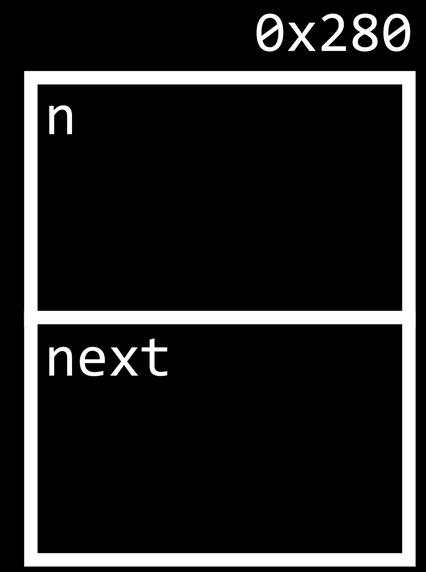
Linked List

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



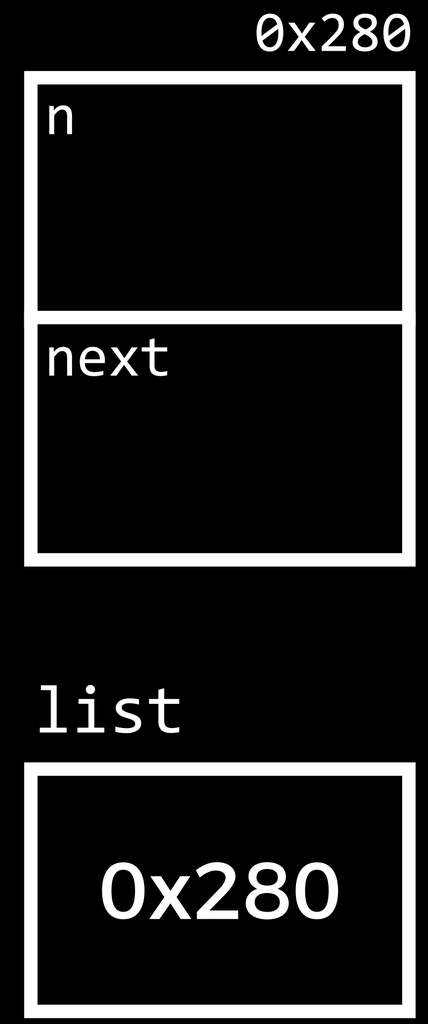
Linked List

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



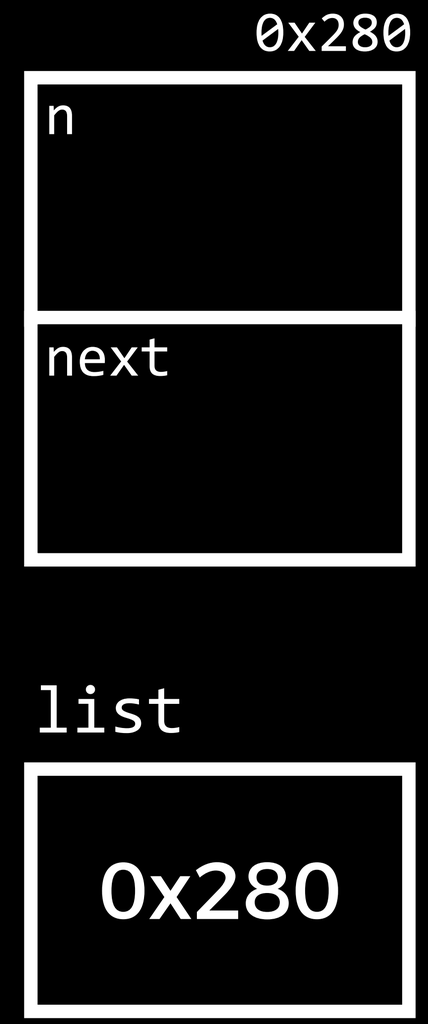
Linked List

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



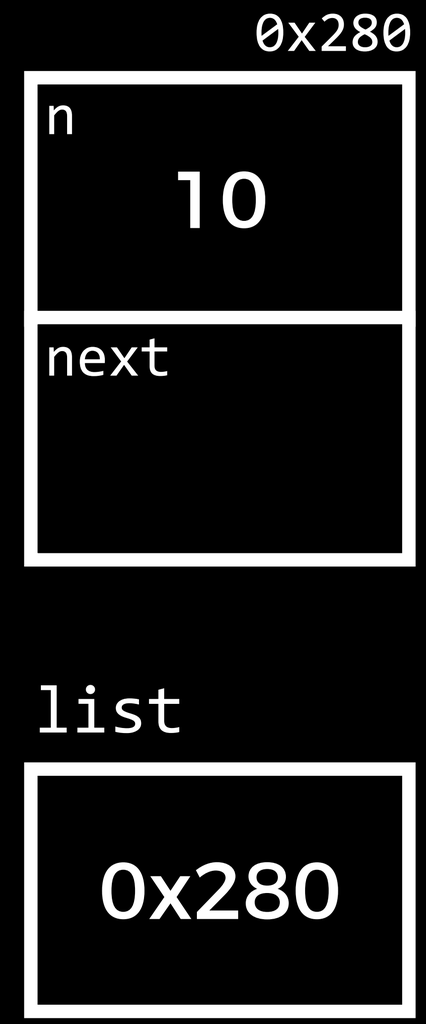
Linked List

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

list->n = 10;
```



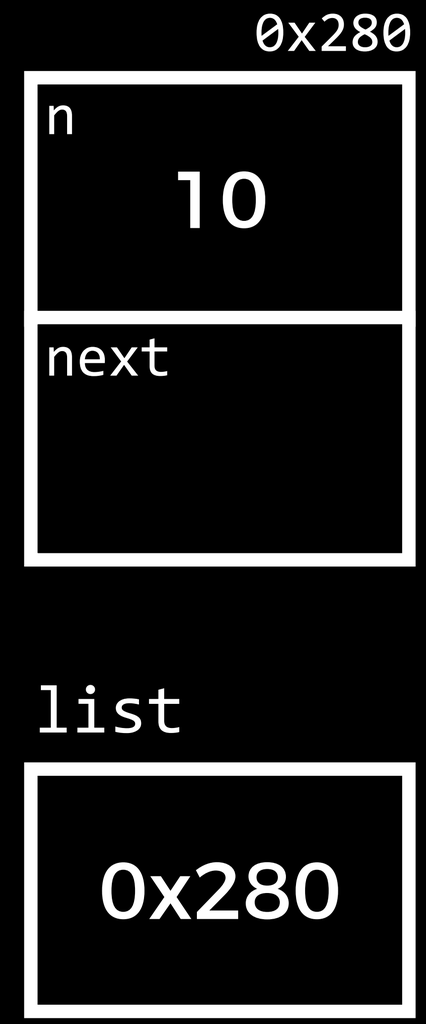
Linked List

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

list->n = 10;
```



Linked List

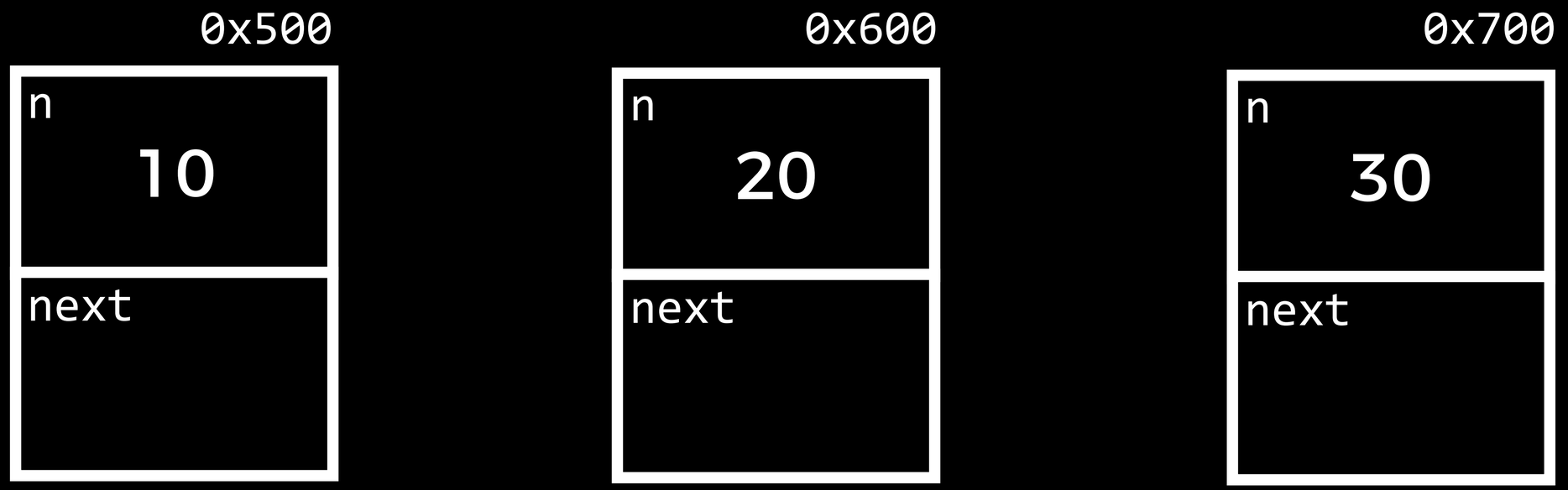
```
node *list = malloc(sizeof(node));

list->n = 10;

list->next = NULL;
```



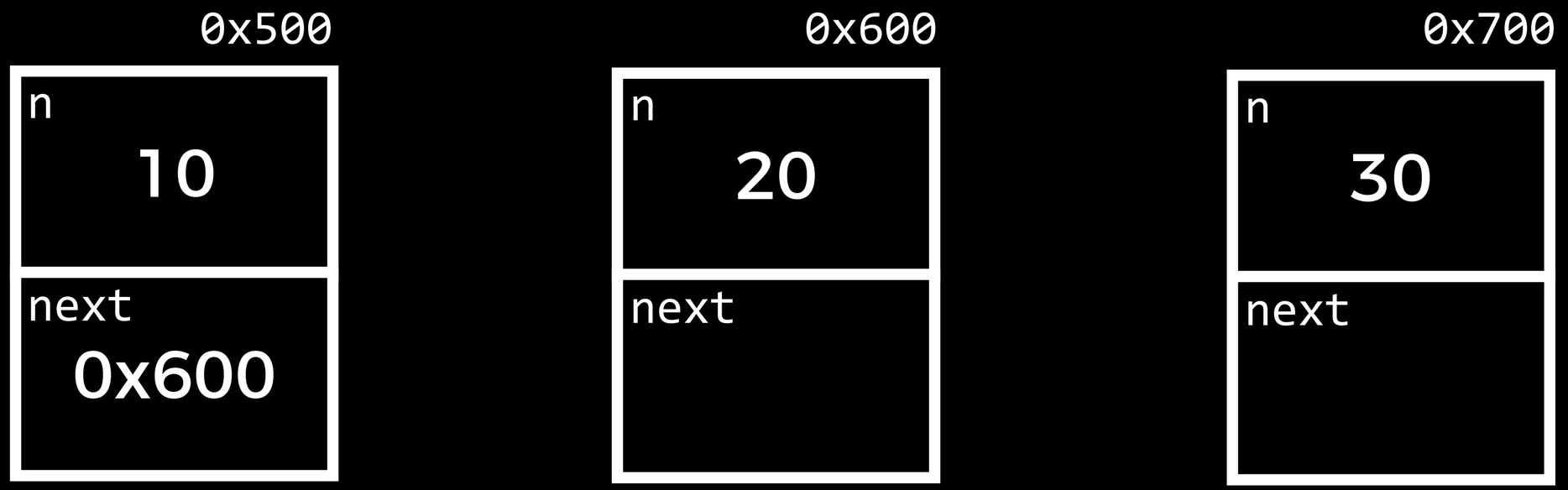
Linked List

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

list->n = 10;

list->next = NULL;
```

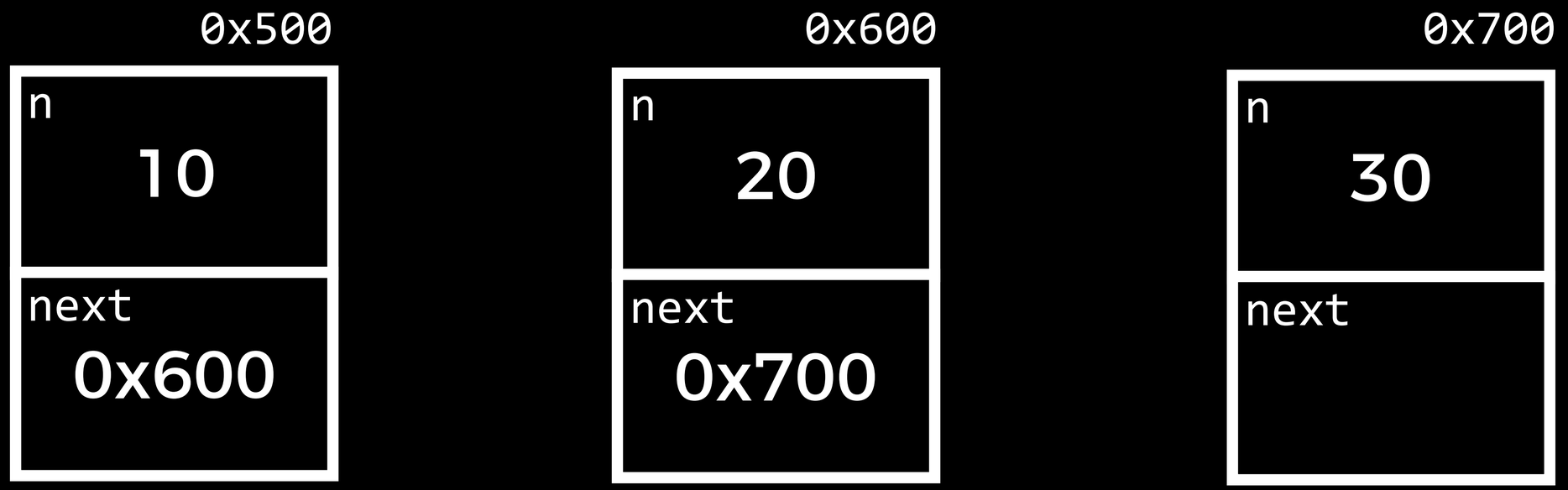
Linked List



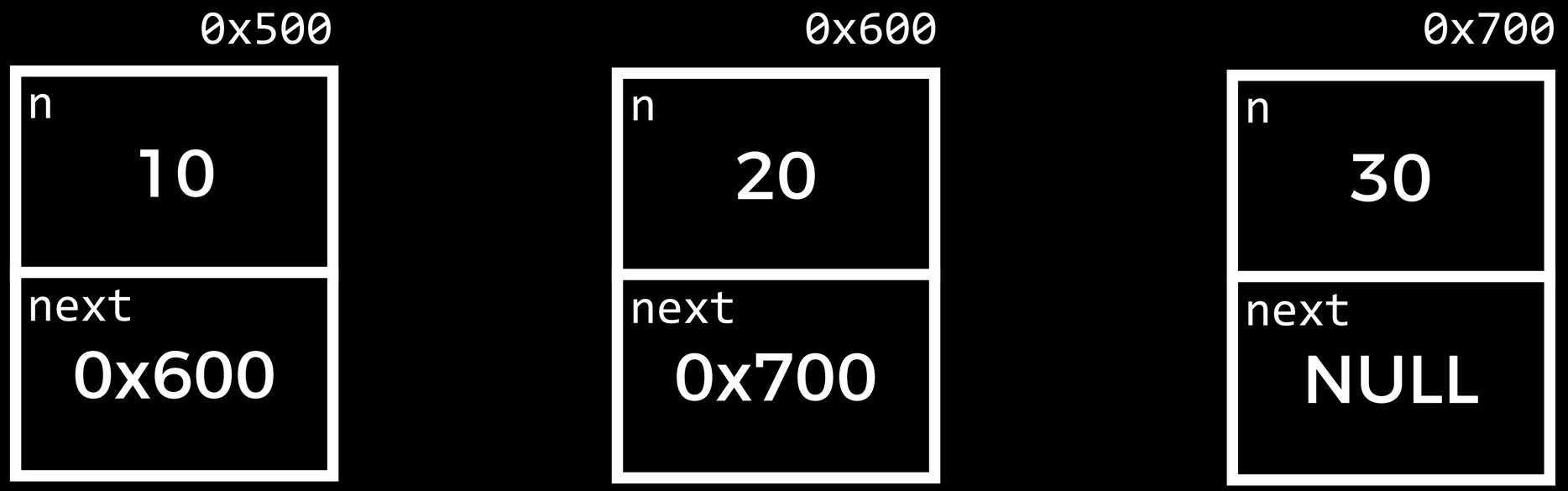
Linked List



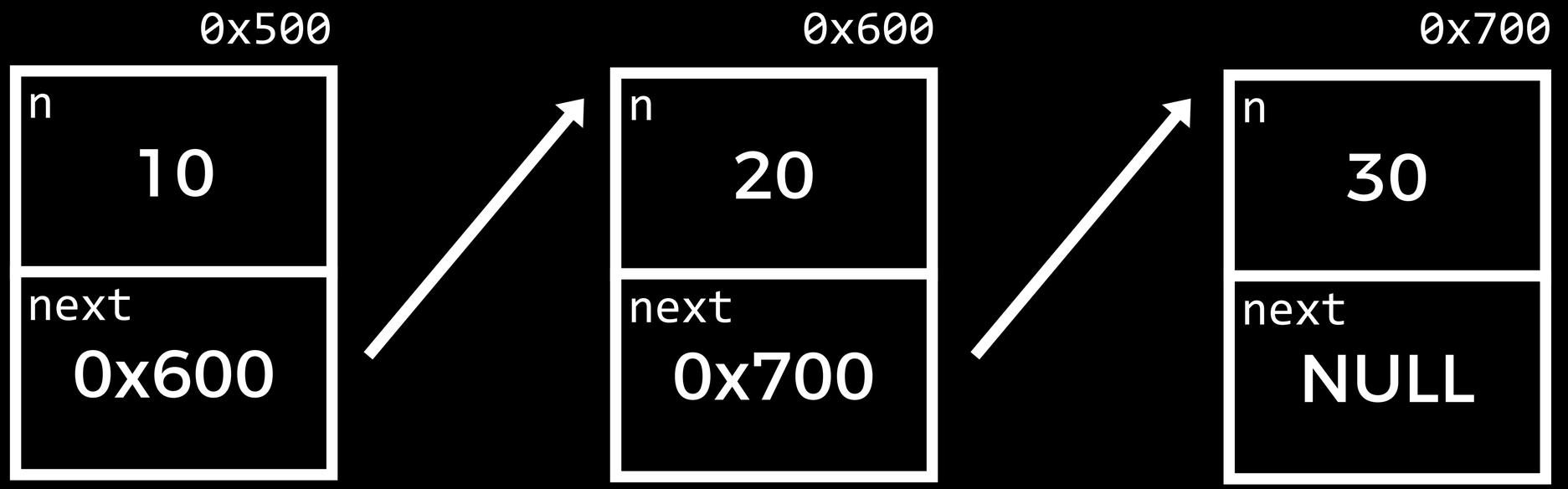
Linked List



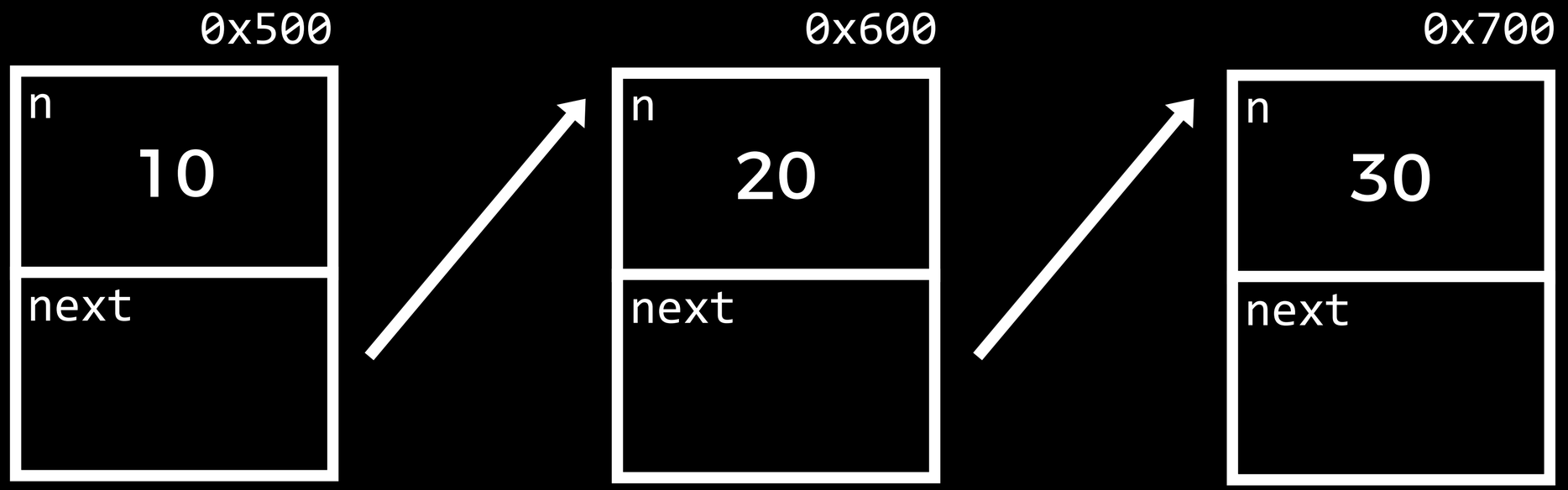
Linked List


0x500
n
10
next
0x600
0x600
n
20
next
0x700
0x700
n
30
next
NULL

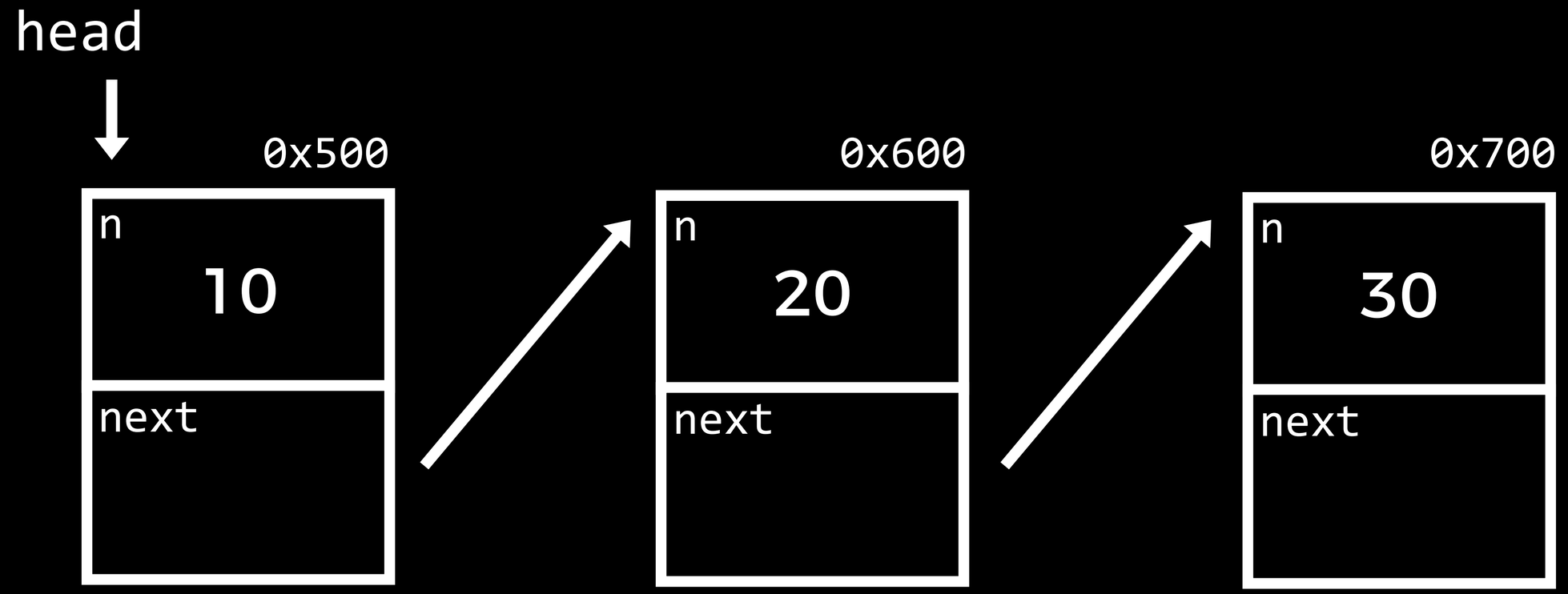
Linked List


0x500
n
10
next
0x600
0x600
n
20
next
0x700
0x700
n
30
next
NULL

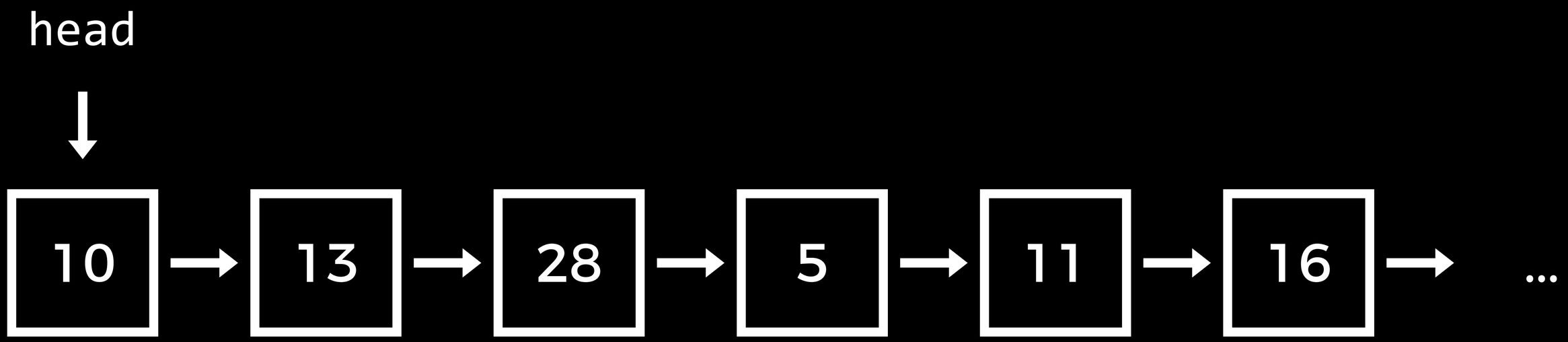
Linked List

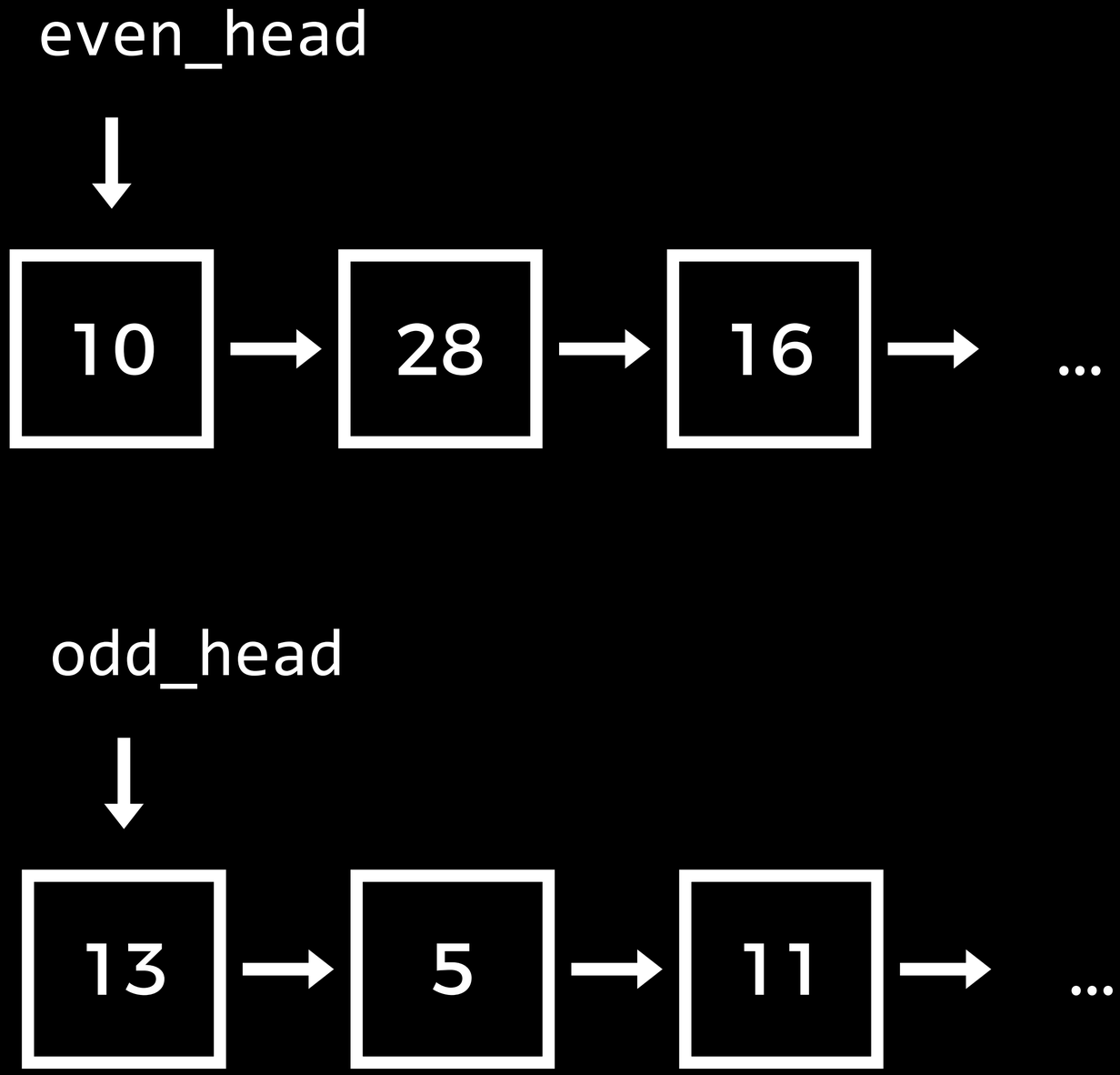


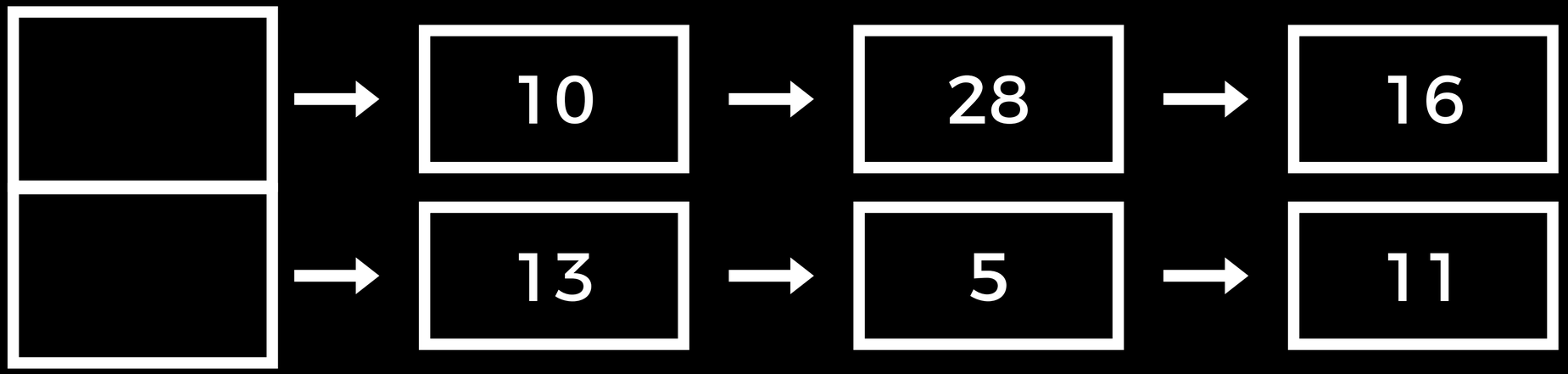
Linked List

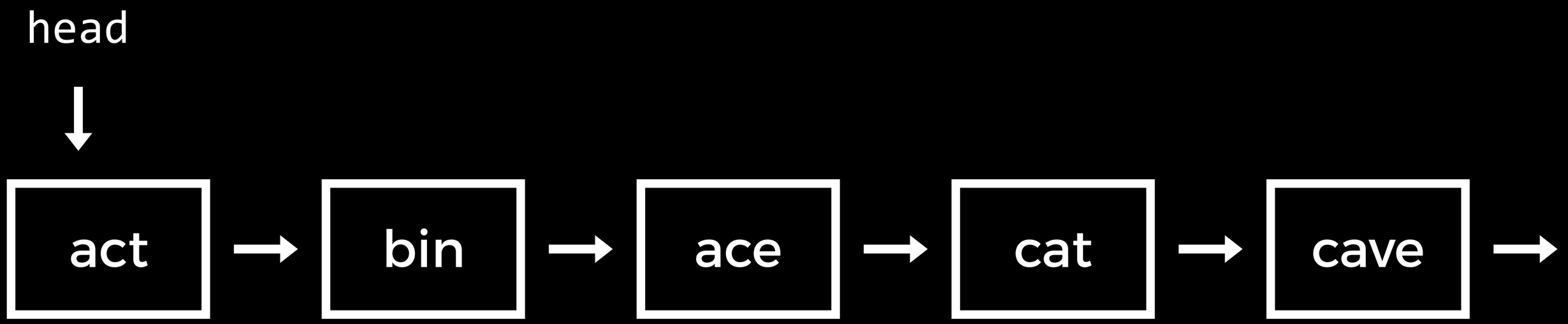


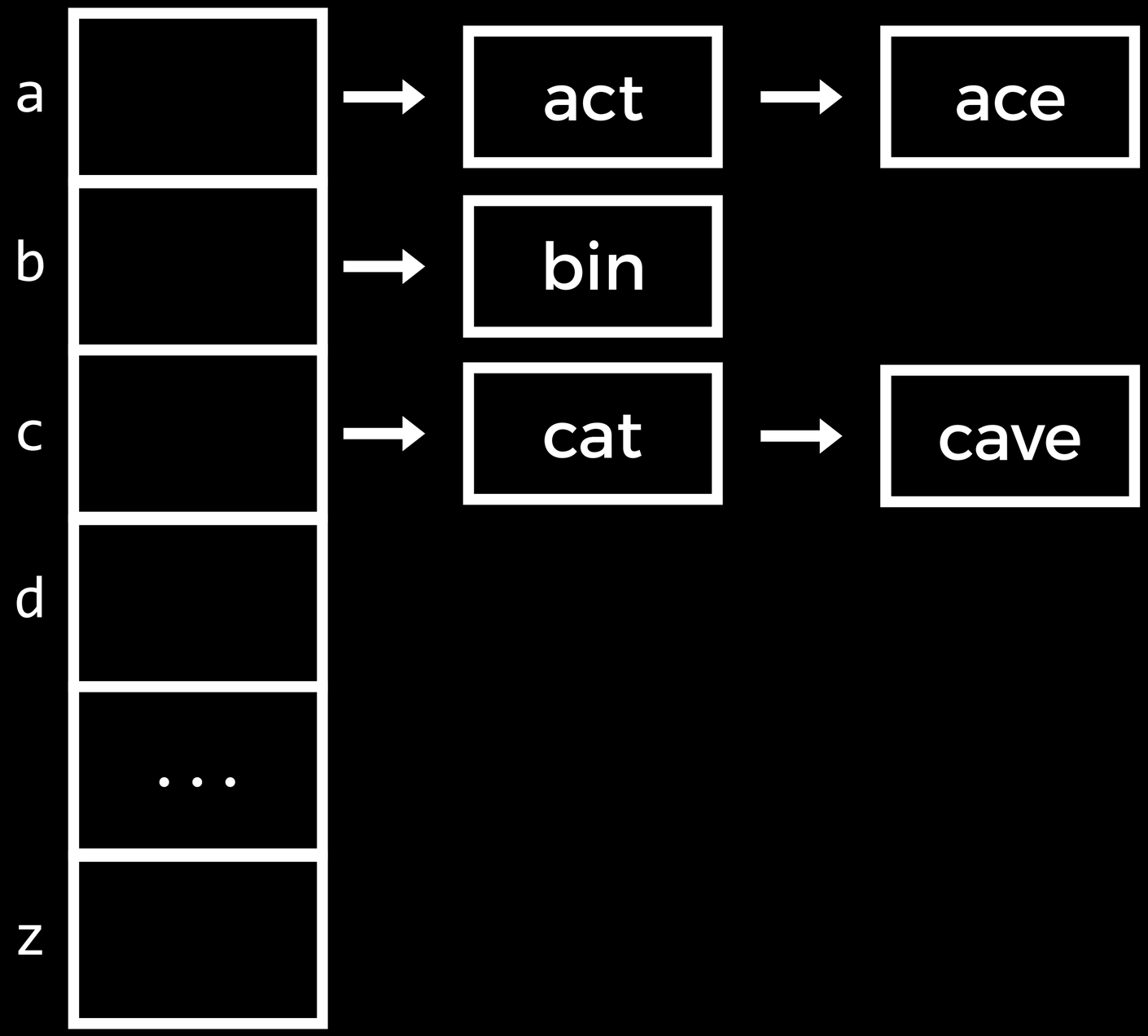
Hash Tables


head
10
13
28
5
11
16
...


even_head
10
28
16
...
odd_head
13
5
11
...


10
28
16
13
5
11

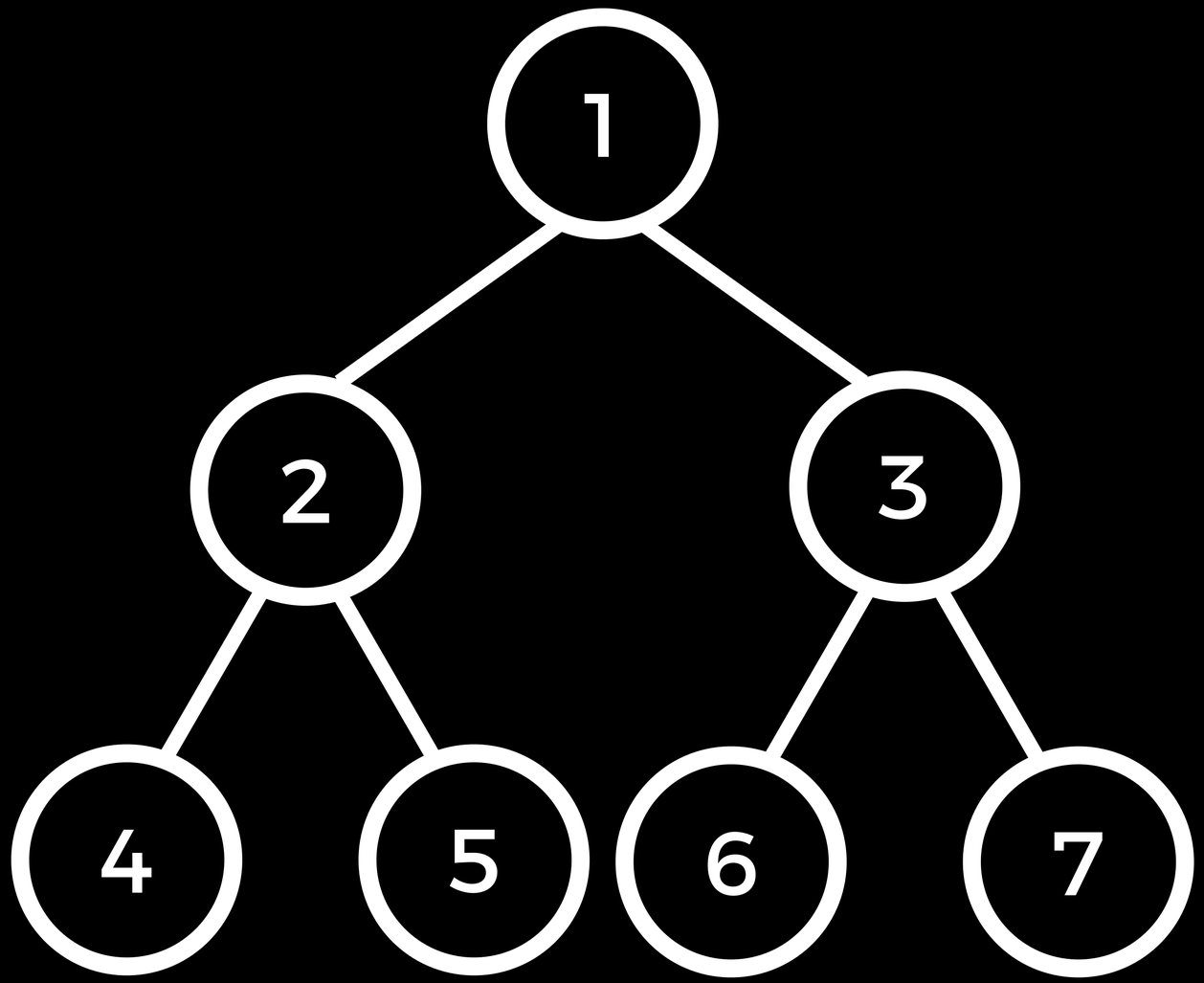

head
act
bin
ace
cat
cave


a
act
ace
b
bin
c
cat
cave
d
. . .
z

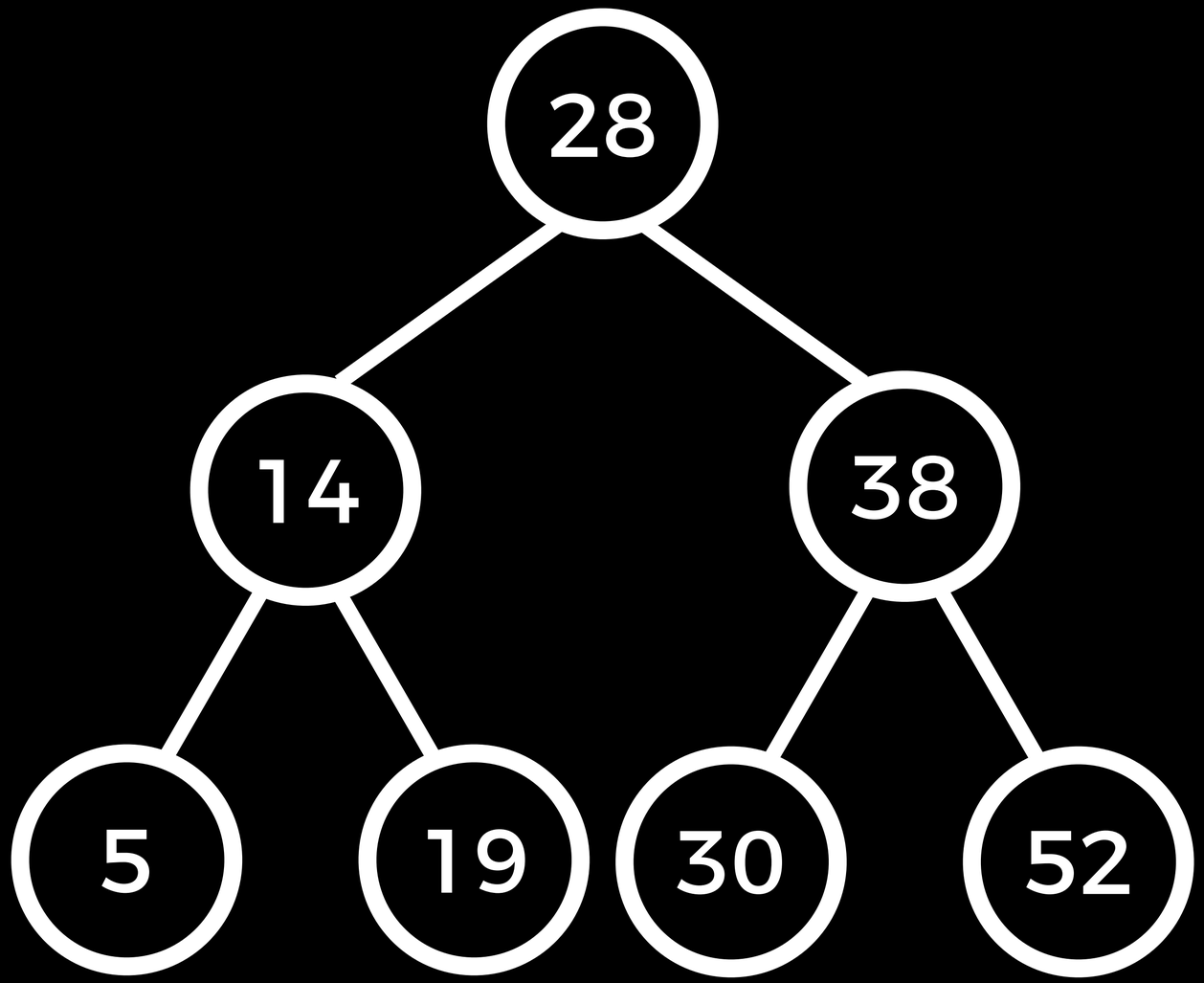
Hash Function

```
unsigned int hash(const char *word)
{
    return tolower(word[0]) - 'a';
}
```

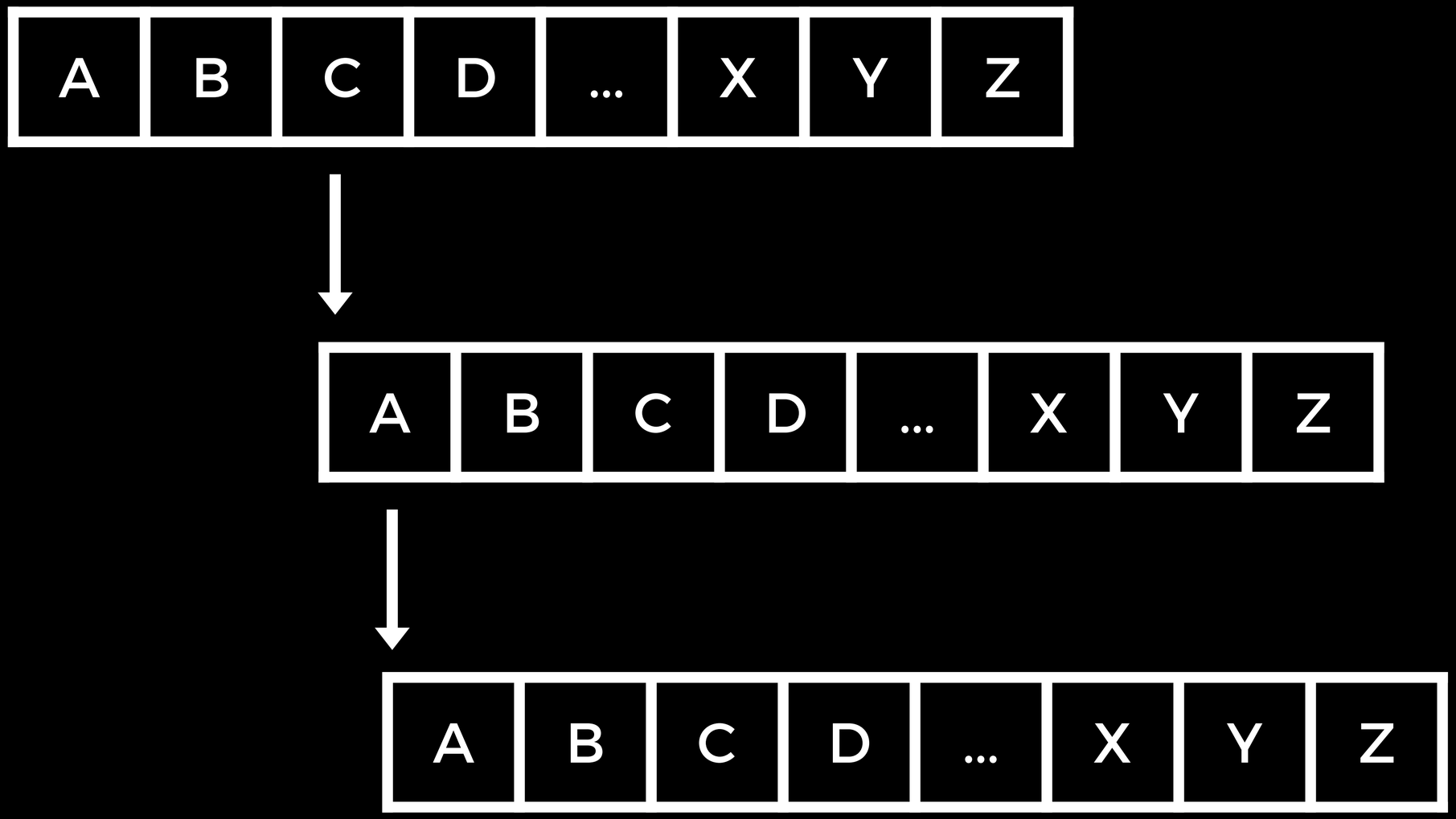
Trees



Binary Search Tree



Tries



Tries

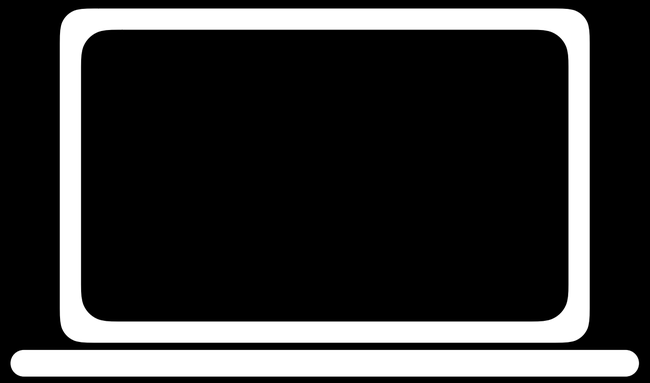
```
typedef struct node
{
    bool is_word;
    struct node *children[27];
}
node;
```

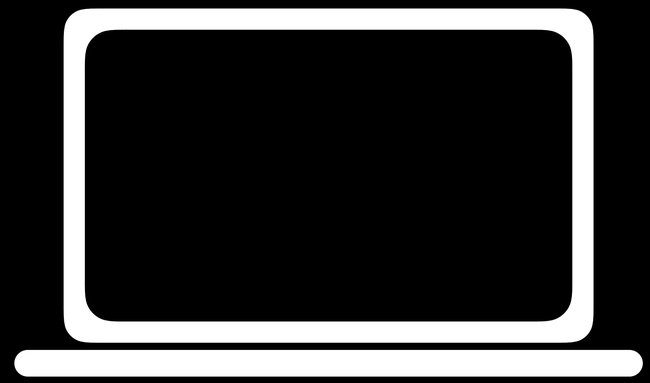
Week 5

HTTP, HTML, CSS

TCP/IP

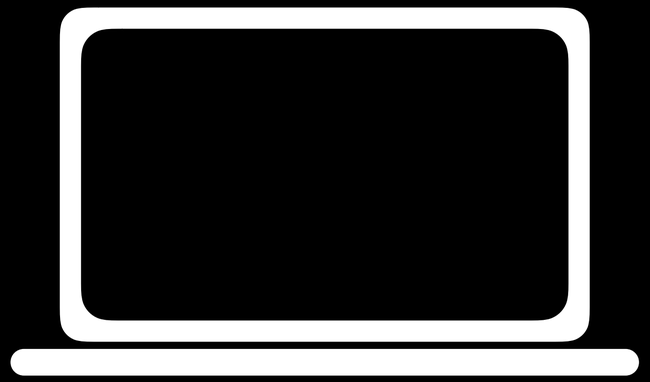
TCP/IP



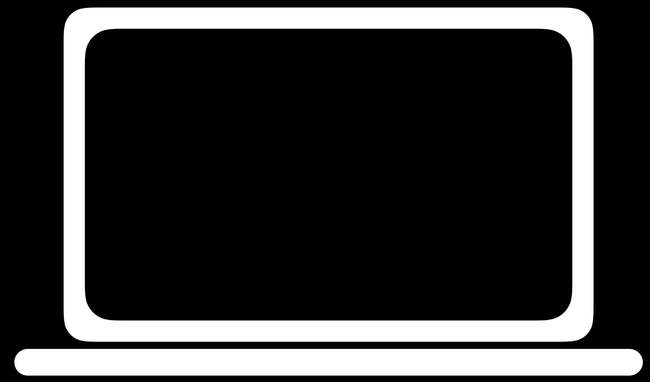


TCP/IP

28.28.28.28



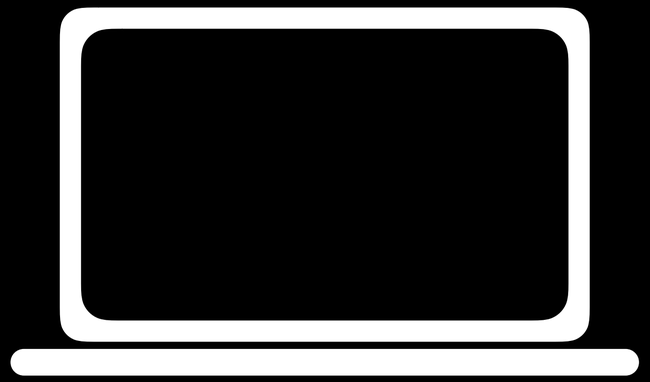
42.42.42.42



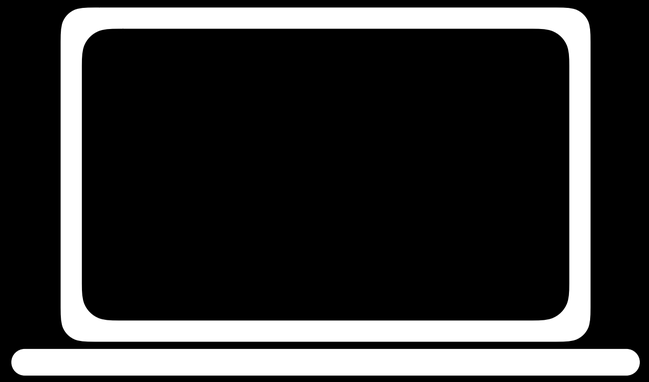
TCP/IP

```
From: 28.28.28.28
To: 42.42.42.42
Port: 80
Packet: 2 of 4
```

28.28.28.28



42.42.42.42



DNS

hostname	ip
google.com	172.217.7.206
harvard.edu	23.22.75.102
yale.edu	104.16.243.4
apple.com	17.172.224.47
github.com	192.30.253.112

HTTP

HTTP

```
GET / HTTP/1.1
Host: www.harvard.edu
```

HTTP Status Codes

- 200
- 301
- 403
- 404
- 500
- ...

HTML

HTML

```
<!DOCTYPE html>

<html>
    <head>
        <title>hello!</title>
    </head>
    <body>
        hello, world!
    </body>
</html>
```

CSS

CSS

```
element
{
    property: value;
}
```

CSS

```
#id
{
    property: value;
}
```

CSS

```
.class
{
    property: value;
}
```

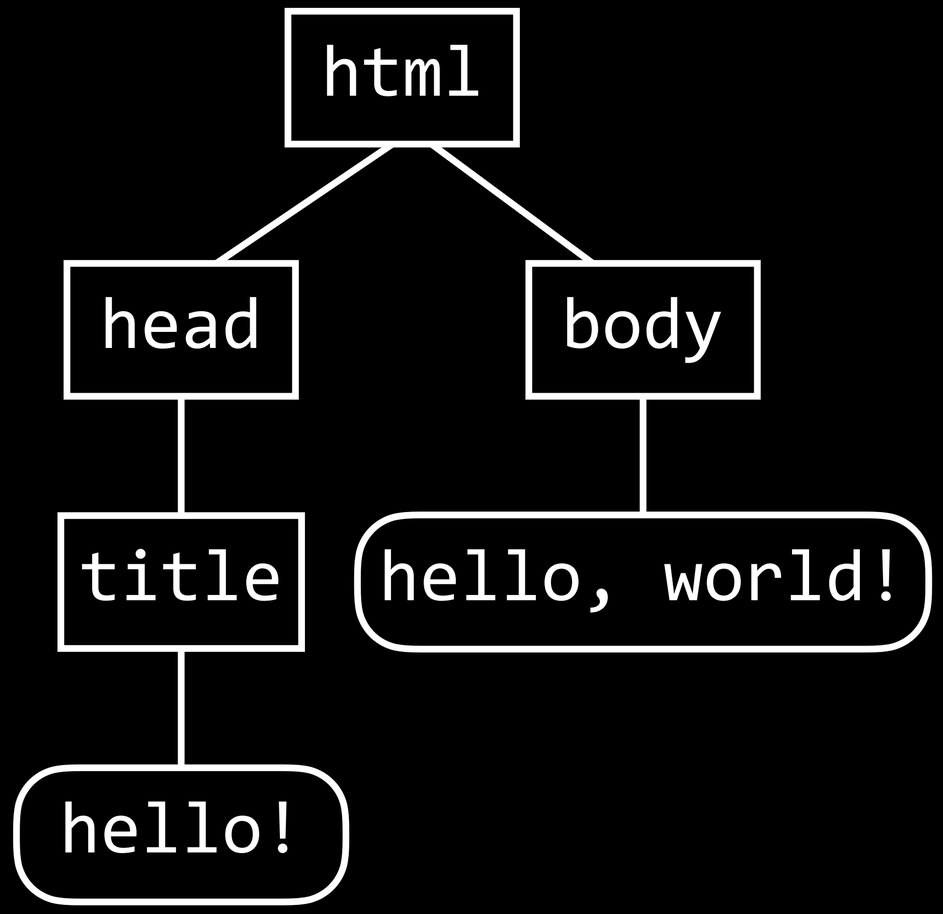
JavaScript

DOM

```
<!DOCTYPE html>
<html>
    <head>
        <title>hello!</title>
    </head>
    <body>
        hello, world!
    </body>
</html>
```

DOM

```
<!DOCTYPE html>
<html>
    <head>
        <title>hello!</title>
    </head>
    <body>
        hello, world!
    </body>
</html>
```



Week 6

Python

Variables

```
x = 28
y = "Hello"
z = True
```

Print

```
print("Hello!")
print("Hello!", end=" ")
print(x)
print("Hello,", name)
```

Conditions

```
if x > 0:
    print("Positive")
else:
    print("Not positive")
```

Conditions

```
if x > 0:
    print("Positive")
elif x < 0:
    print("Negative")
else:
    print("Zero")
```

Lists

```
days = ["Sunday", "Monday", "Tuesday"]
```

Loops

```
for i in range(5):
    print(i)
```

Loops

```
for item in items:
    print(item)
```

Functions

```
def add(a, b):
    return a + b
```

Data Structures

```
list
tuple
set
dict
```

Week 7

Web Programming

MVC

- Model
- View
- Controller

Flask

```
from flask import Flask, render_template

app = Flask(__name__)

@app.route("/")
def index():
    return render_template("index.html")
```

Week 8

SQL

Database

Table

Table

Table

Table

SQL Data Types

- INTEGER: smallint, integer, bigint
- NUMERIC: boolean, date, datetime, numeric(scale, precision), time, timestamp
- REAL: real, double precision
- TEXT: char(n), varchar(n), text

SQL

```
CREATE TABLE registrants (
    id INTEGER PRIMARY KEY,
    name VARCHAR(255),
    dorm VARCHAR(255)
)
```

SQL

```
INSERT INTO registrants (id, name, dorm)
VALUES (1, 'David', 'Matthews')
```

SQL

```
SELECT * FROM registrants
```

SQL

```
SELECT name, dorm FROM registrants
```

SQL

```
SELECT name, dorm FROM registrants
WHERE dorm = 'Matthews'
```

SQL

```
UPDATE registrants
SET name = 'David Malan'
WHERE id = 1
```

SQL

```
DELETE FROM registrants WHERE id = 1
```

Multiple Tables

- Foreign Keys
- Joining Tables

SQL

- Race Conditions
- SQL Injection Attacks

CS50 Quiz Preparation