

This is CS50

Week 3

Scan your HUID for attendance at the back table.
Open code.cs50.io and log in!

carterzenke.me/section

Think, Pair, Share

- What are you excited about from this week's lecture?
- What do you want to learn more about?

<https://carterzenke.me/section>

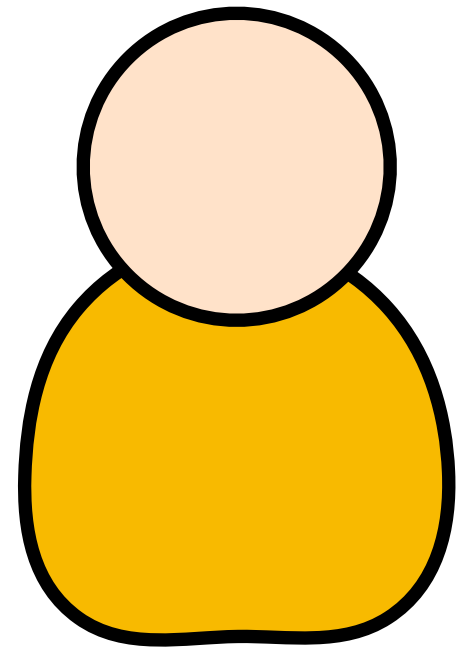
Today

- How can we compare algorithms with O and Ω notation?
- What are **structs**?
- How can we make use of **recursion**?

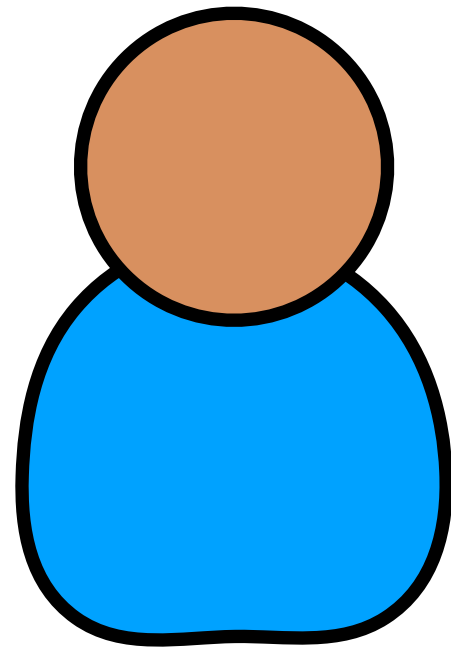
Searching and Sorting

(and O and Ω notation)

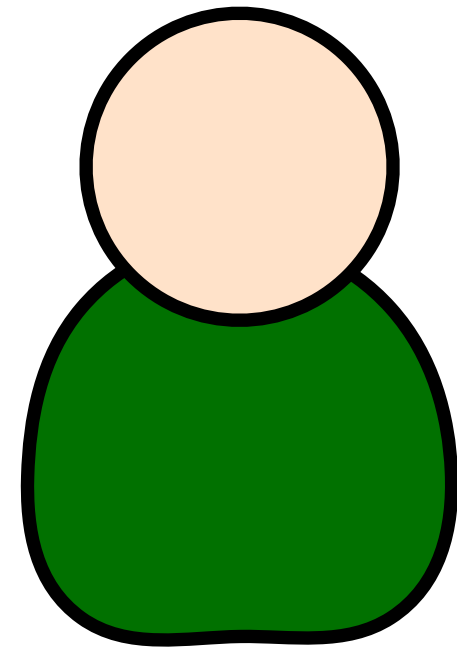
Matthew



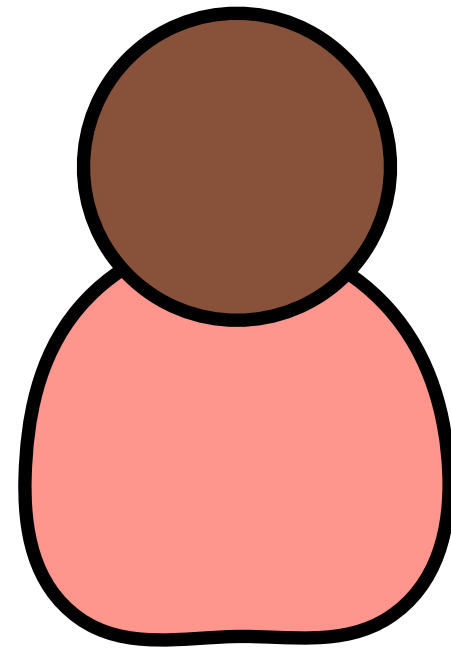
Samia



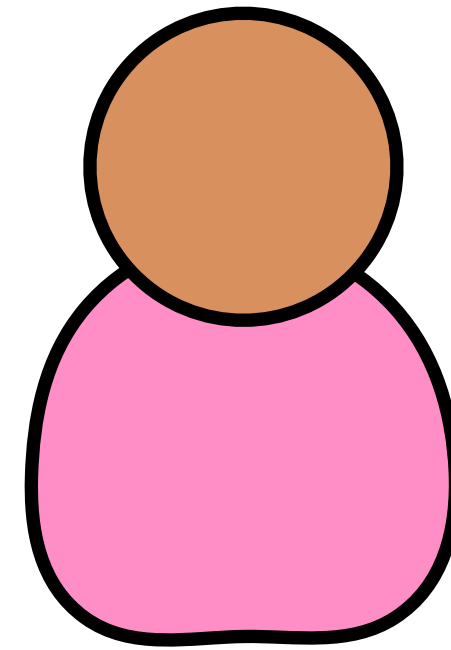
Alyssa



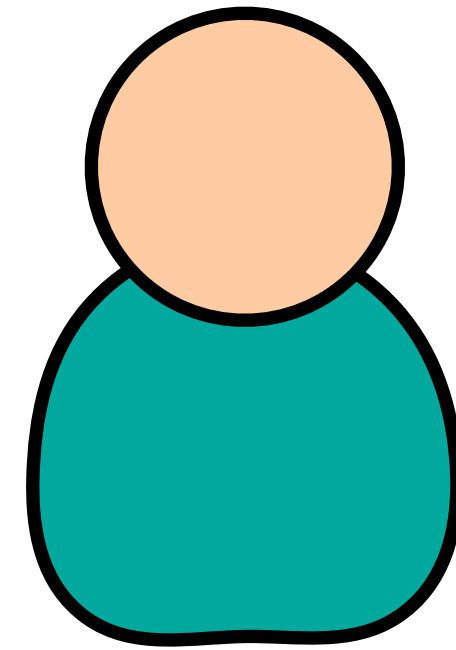
Douglas



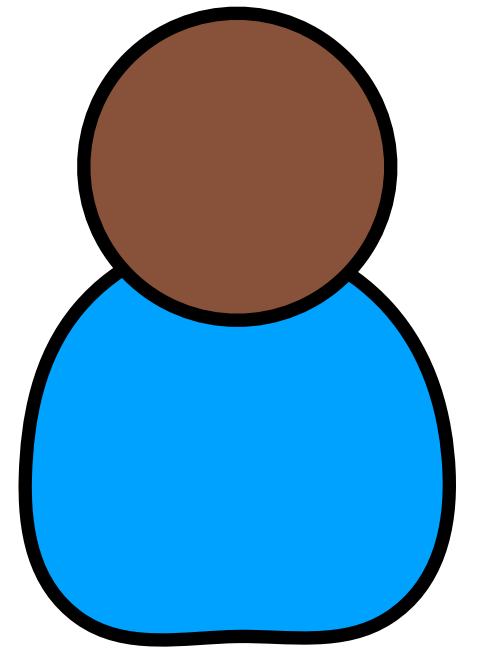
Cecelia



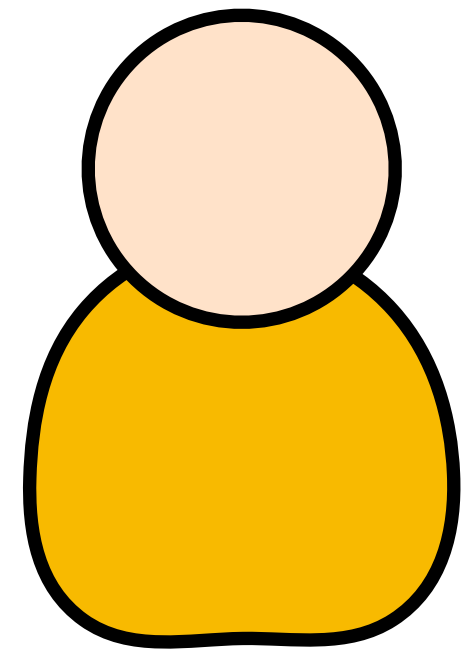
Lucas



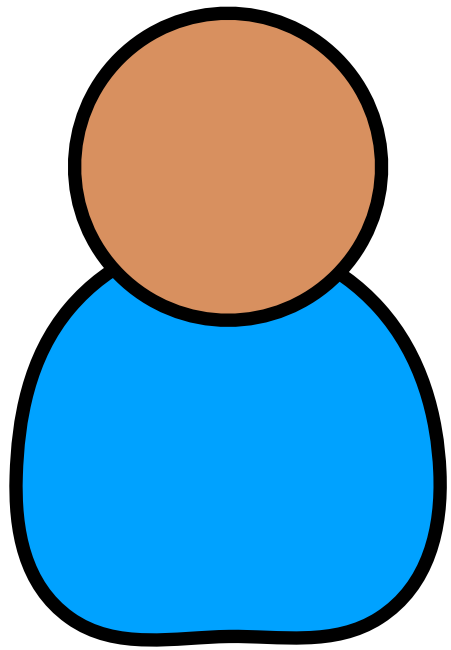
Ramya



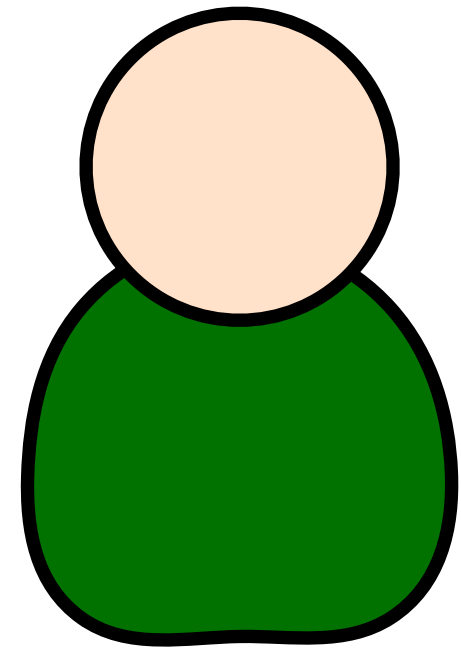
Matthew



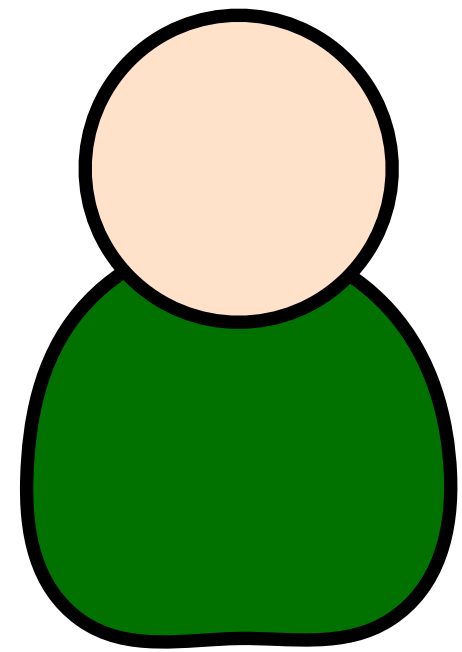
Samia



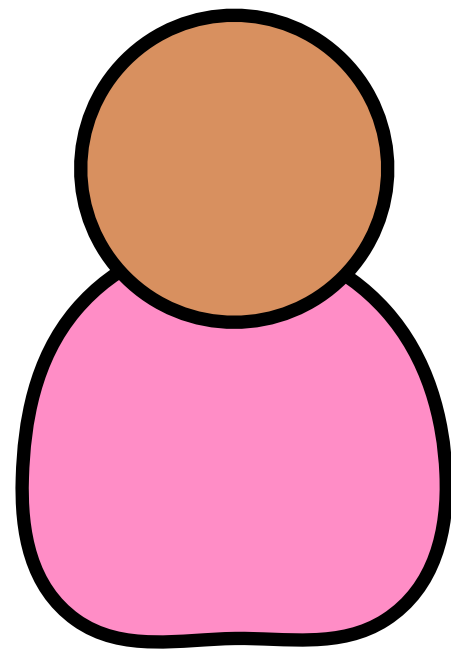
Alyssa



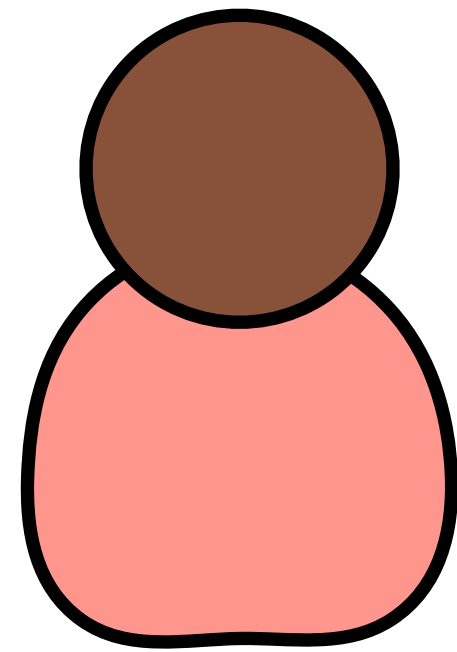
Alyssa



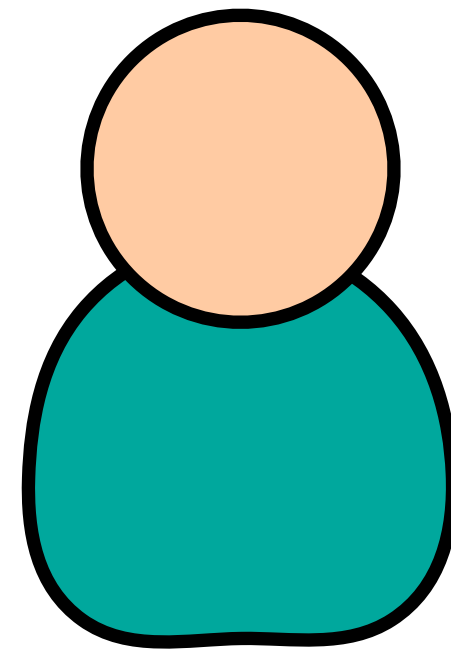
Cecelia



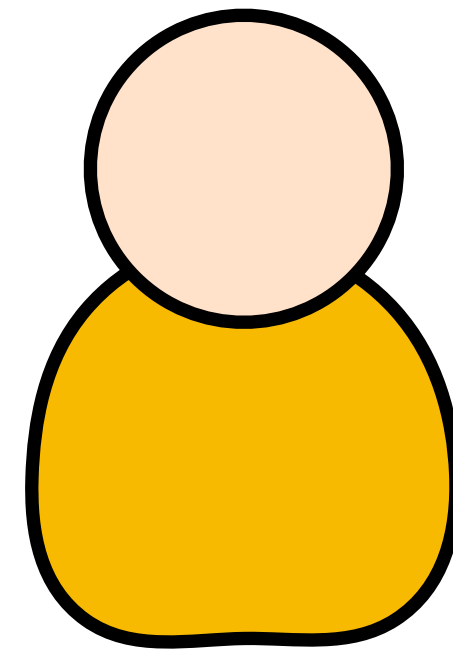
Douglas



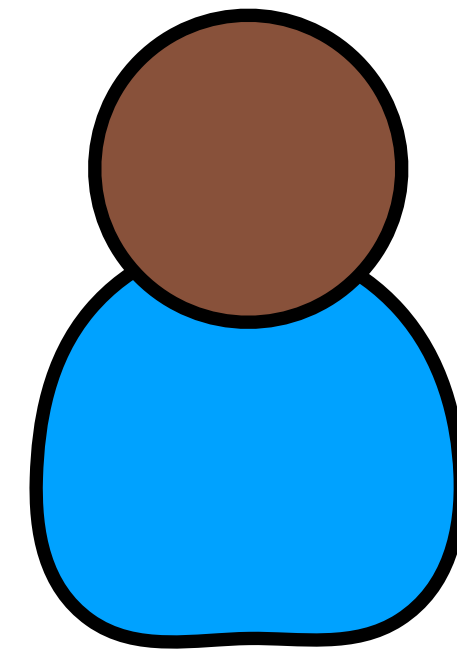
Lucas



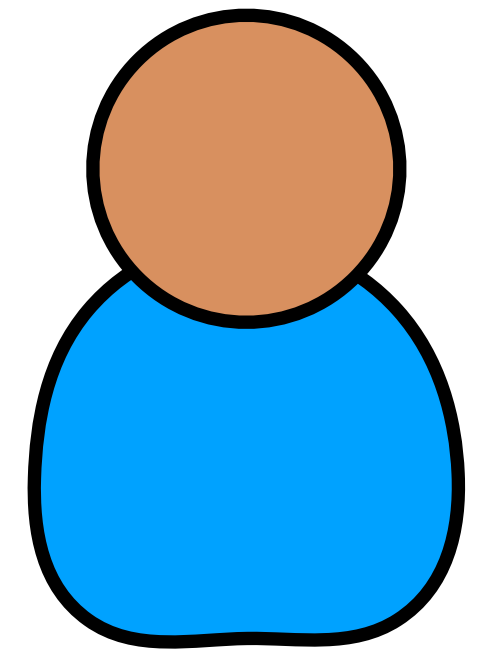
Matthew



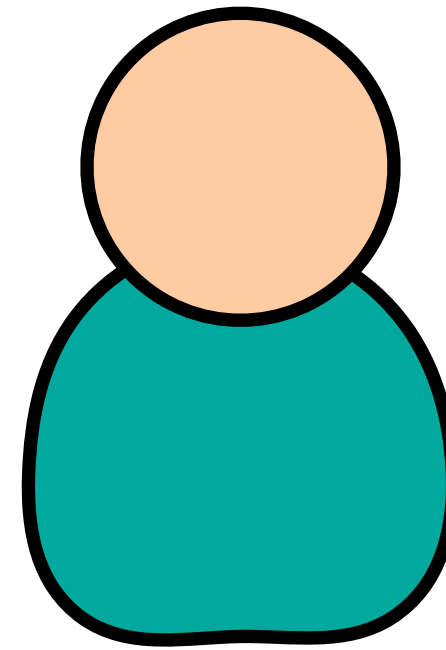
Ramya



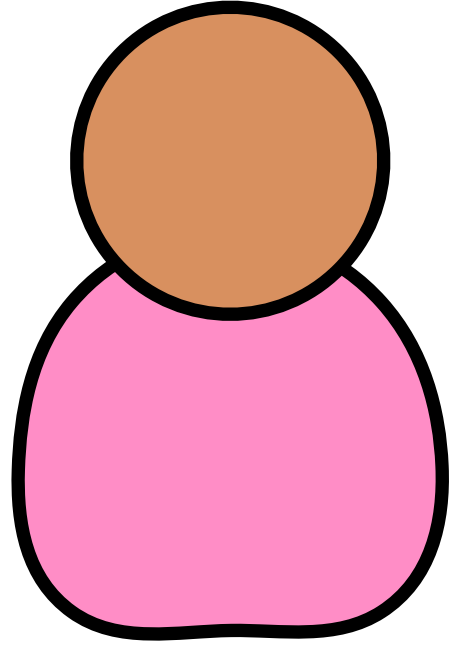
Samia



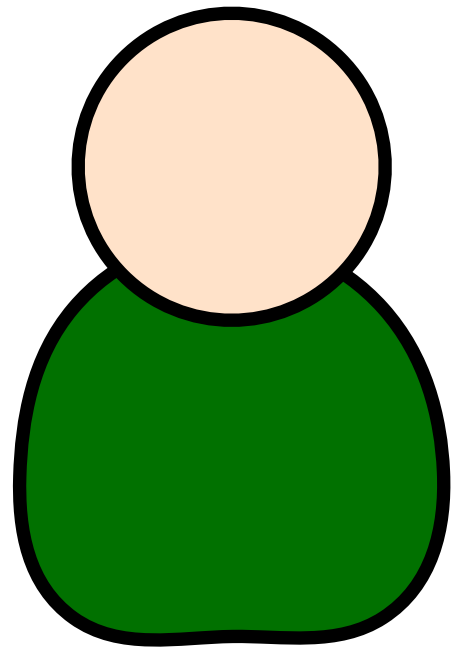
Lucas



Cecelia



Alyssa



How many steps did each algorithm take?

Linear Search

Binary Search

3 steps

3 steps

What's the greatest number of steps
this algorithm will *ever* take?

Linear Search

Binary Search

N

$\log_2(N)$

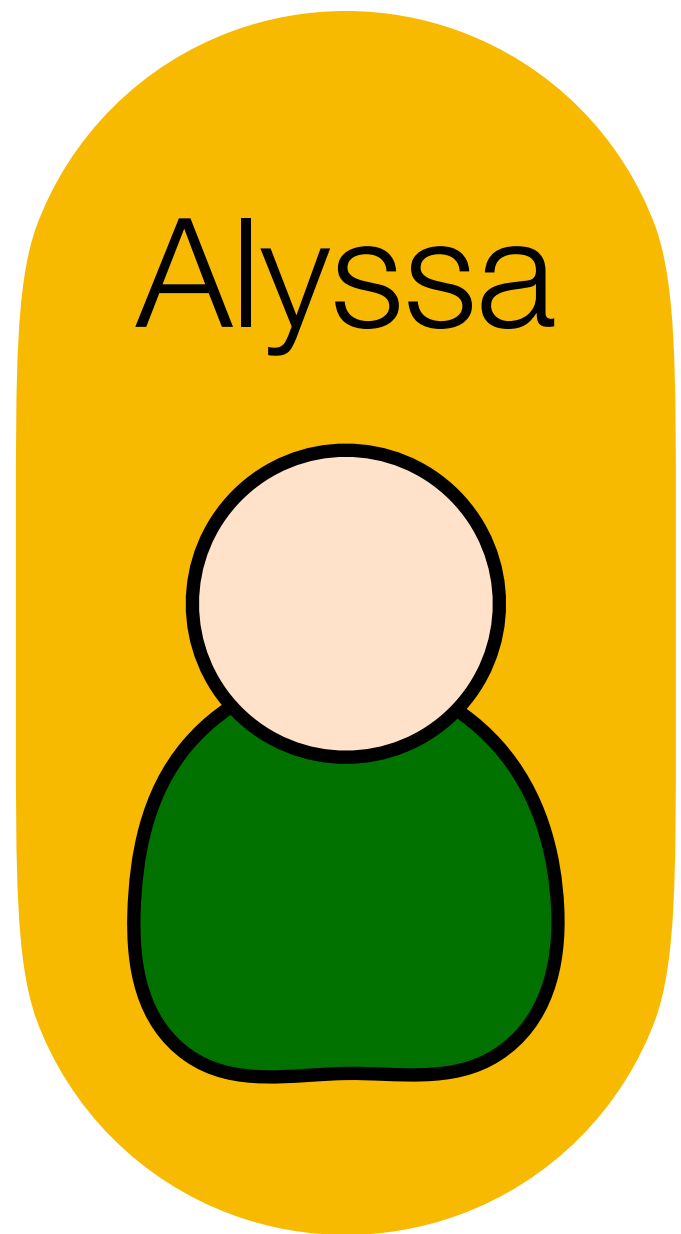
What's (approximately!) the greatest number of steps this algorithm will *ever* take?

Linear Search

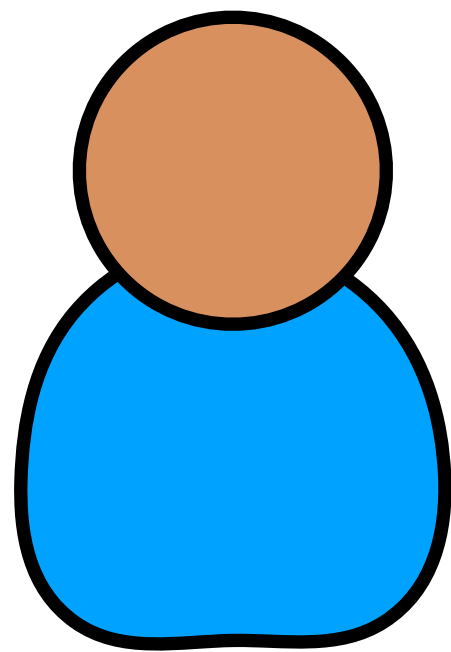
Binary Search

$O(N)$

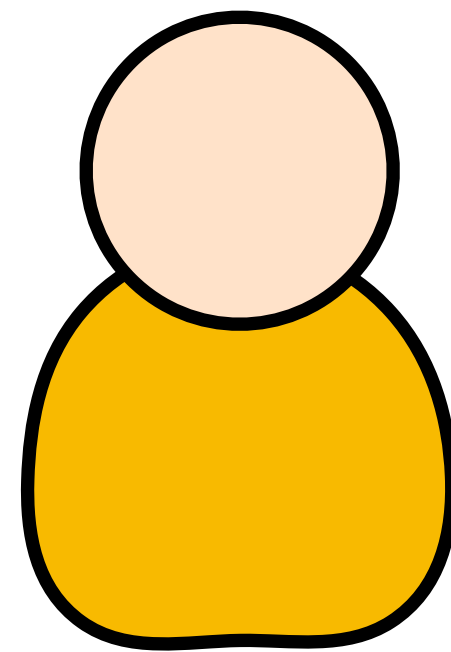
$O(\log(N))$



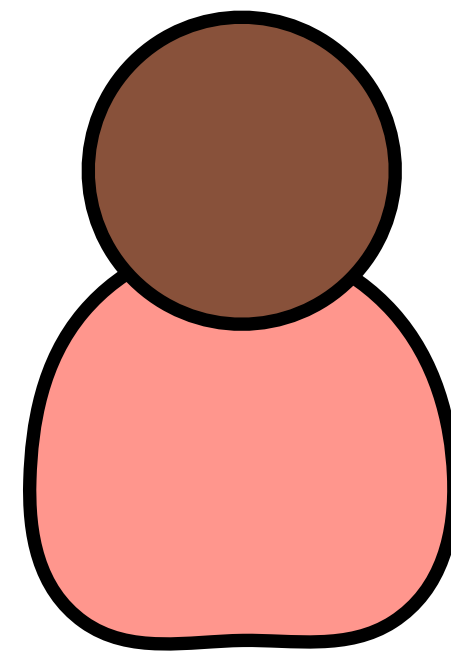
Alyssa



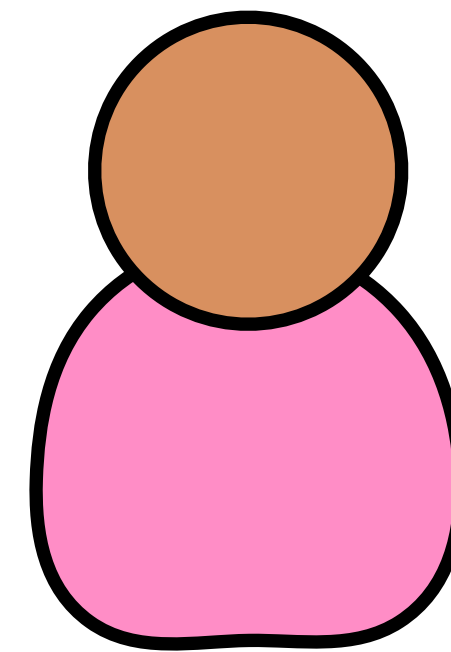
Samia



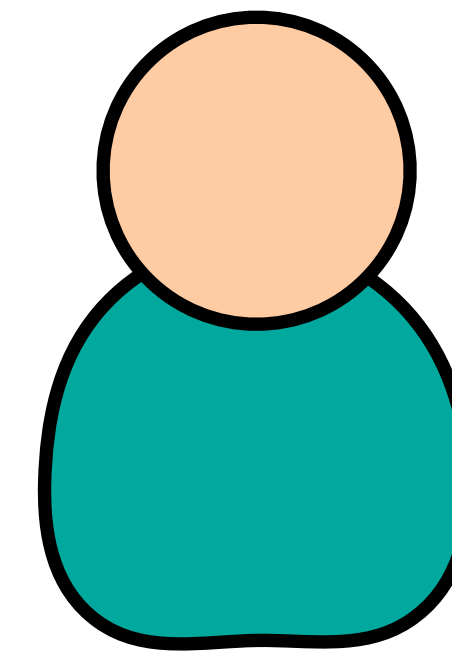
Matthew



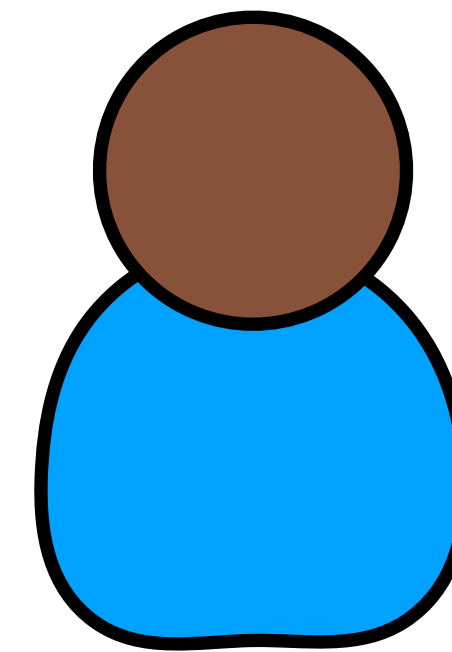
Douglas



Cecelia



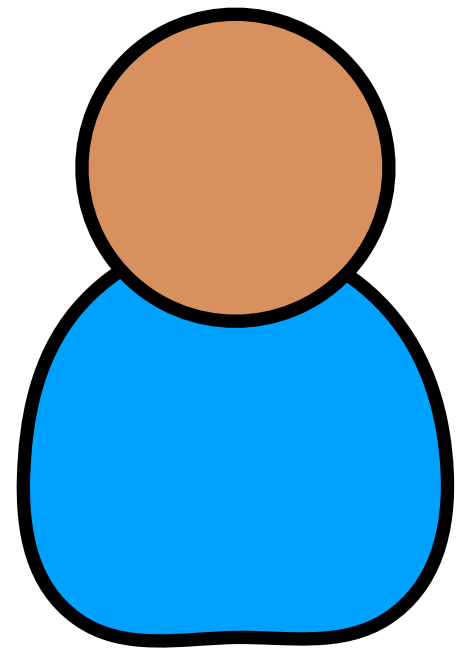
Lucas



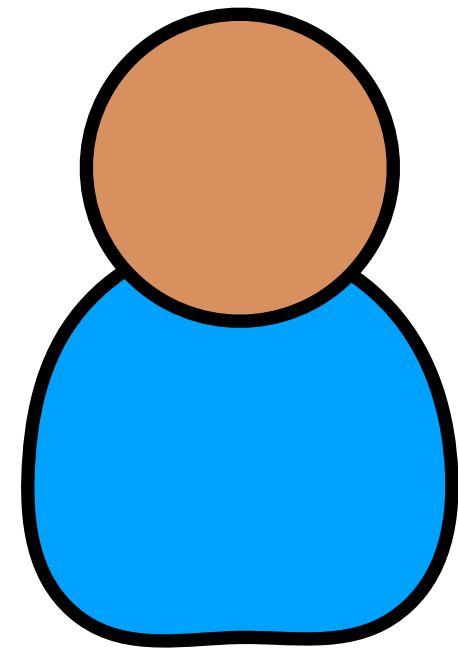
Ramya

Linear Search

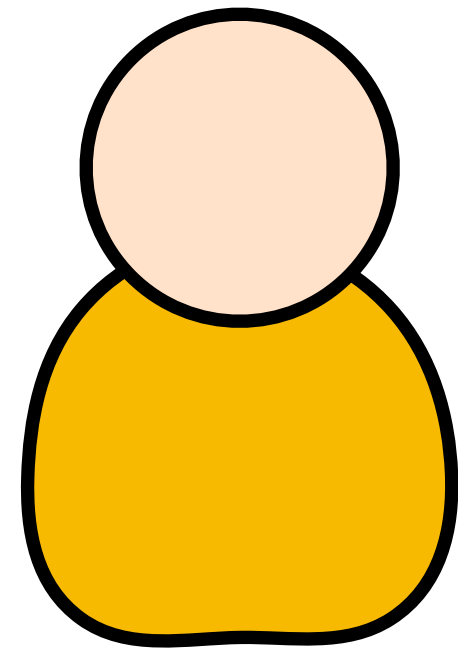
Aaron



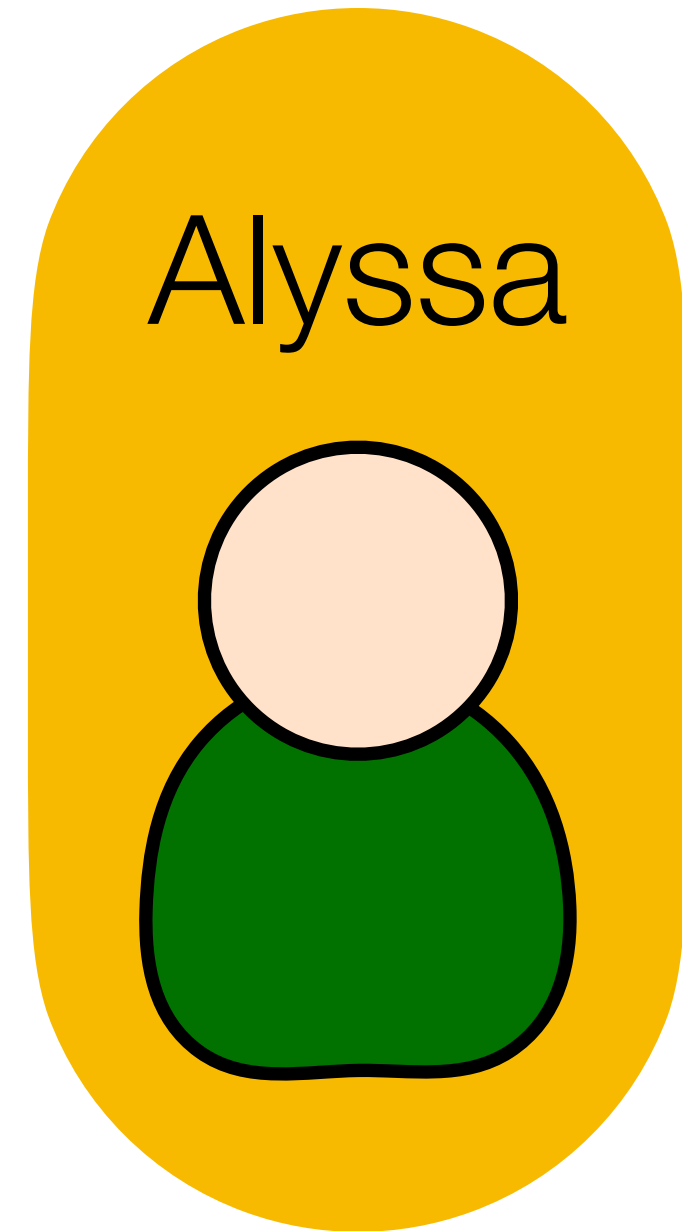
Amulya



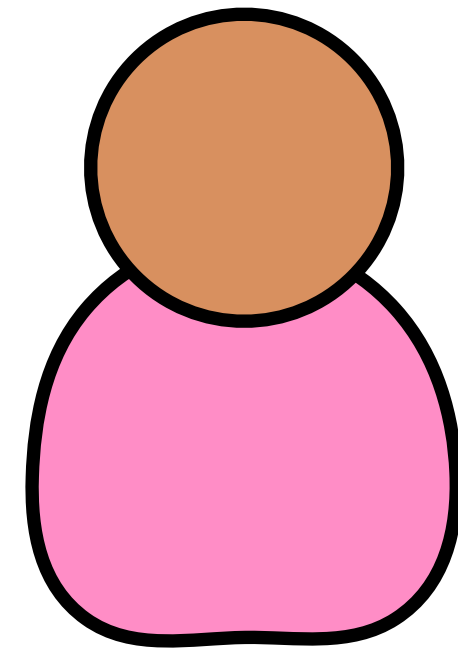
Alex



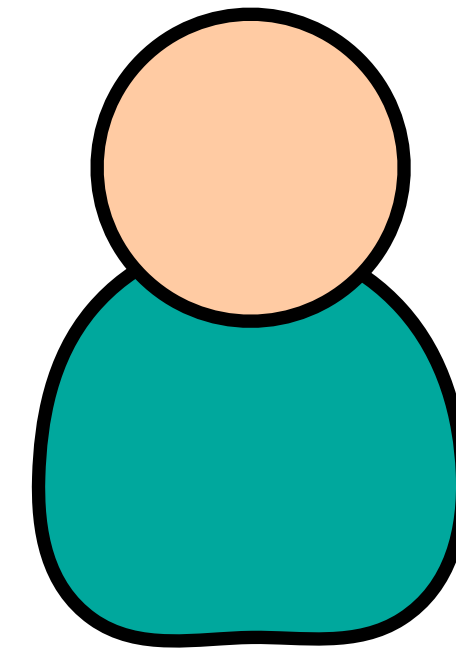
Alyssa



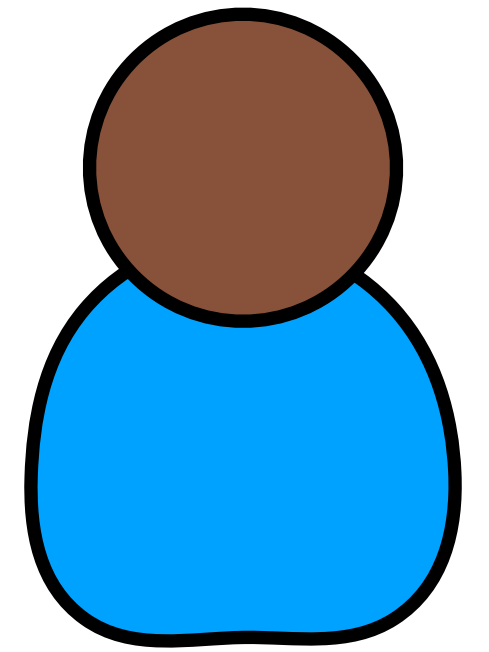
Cecelia



Lucas



Ramya



Binary Search

How many steps did each algorithm take?

Linear Search

Binary Search

1 step

1 step

What's the fewest number of steps
this algorithm will *ever* take?

Linear Search

Binary Search

1

1

What's (approximately!) the fewest number of steps this algorithm will *ever* take?

Linear Search

Binary Search

$\Omega(1)$

$\Omega(1)$

Thought Question

- Suppose that you create a new algorithm and assess its runtime.
- The fewest steps this algorithm will *ever* take is 2, and only 2.
- What is the Ω notation for this algorithm?

Sort

Algorithm	O	Ω
Merge Sort	$O(N \log(N))$	$\Omega(N \log(N))$
Selection Sort	$O(N^2)$	$\Omega(N^2)$
Bubble Sort	$O(N^2)$	$\Omega(N)$

Algorithm

reversed50000.txt

sorted50000.txt

Sort1

Sort2

Sort3

Structs

(and Plurality elections)




```
typedef struct
{
    string name;
    int votes;
}
candidate;
```

```
typedef struct
{
    string name;
    int votes;
}
candidate;
```

Create a new "type",
which holds a collection
of other basic types.

```
typedef struct
{
    string name;
    int votes;
}
candidate;
```

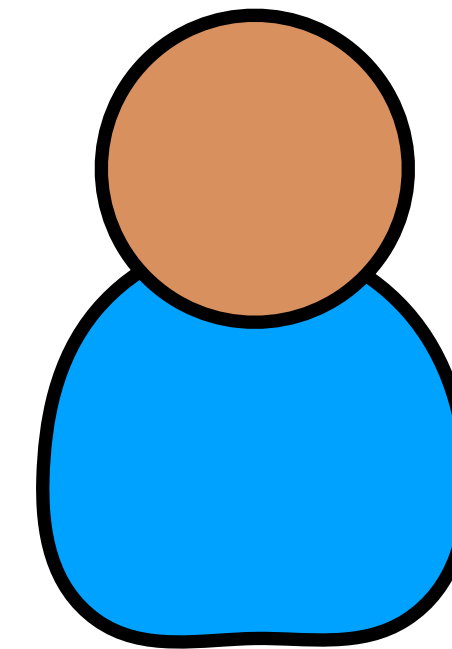
Give the struct a name that can be re-used in the rest of the file.

```
typedef struct
{
    string name;
    int votes;
}
candidate;
```

Known as a structure's
members.


```
typedef struct
{
    string name;
    int votes;
}
candidate;

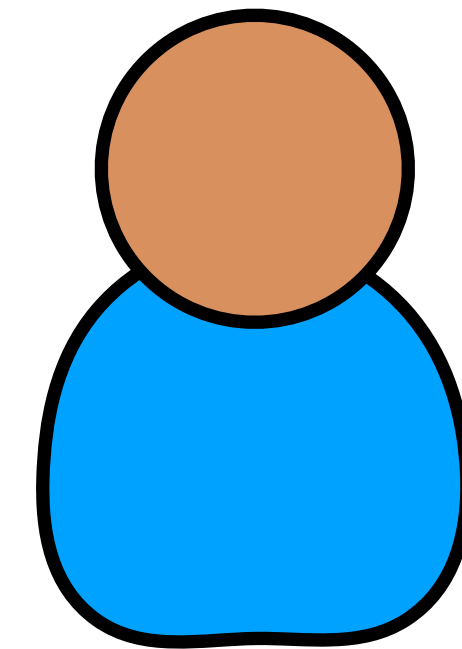
candidate president;
```



```
typedef struct
{
    string name;
    int votes;
}
candidate;

candidate president;
president.name = "Samia";
```

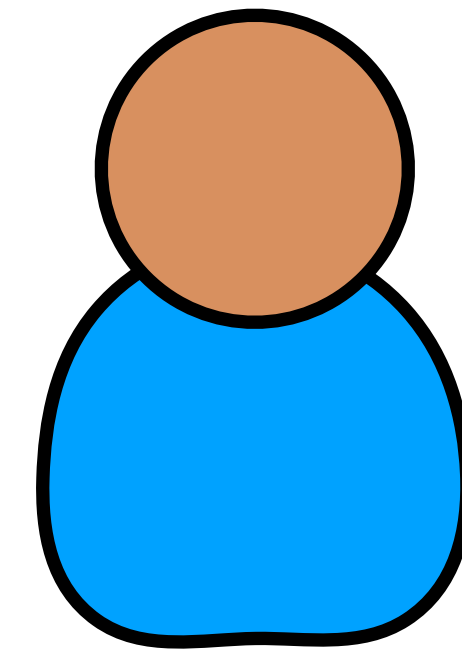
Samia



```
typedef struct
{
    string name;
    int votes;
}
candidate;

candidate president;
president.name = "Samia";
president.votes = 10;
```

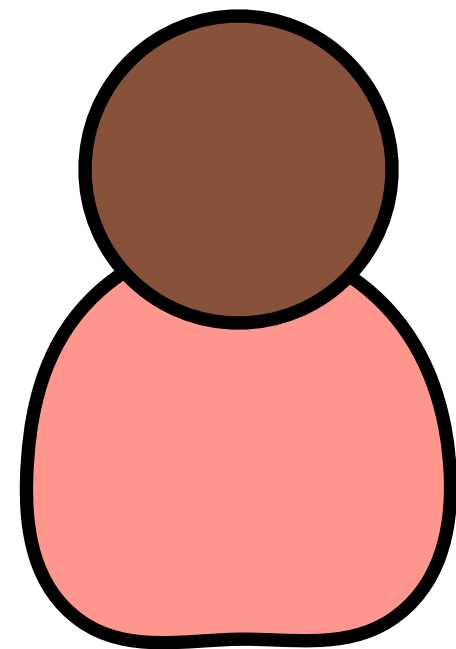
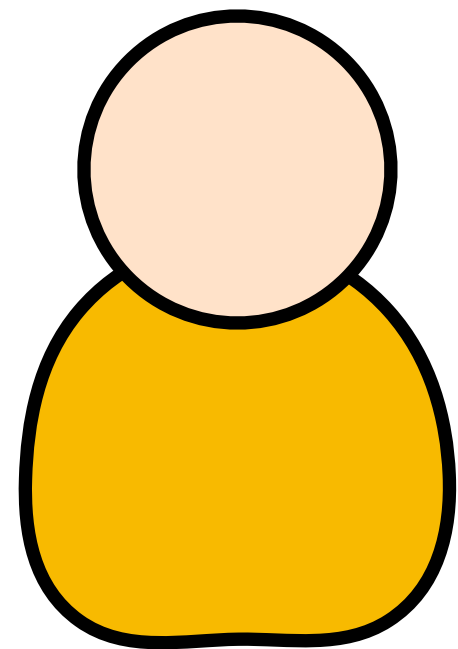
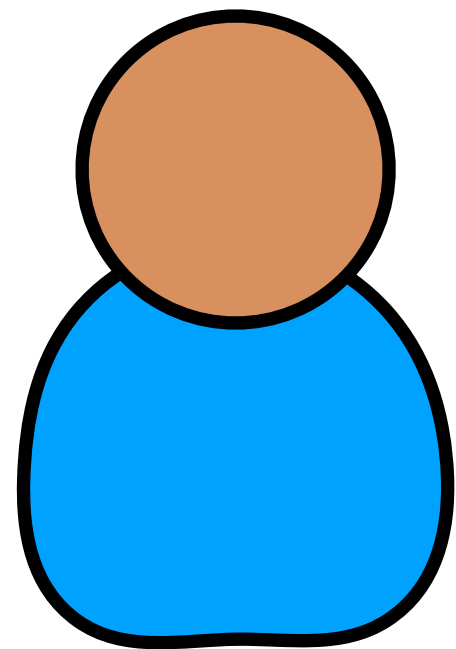
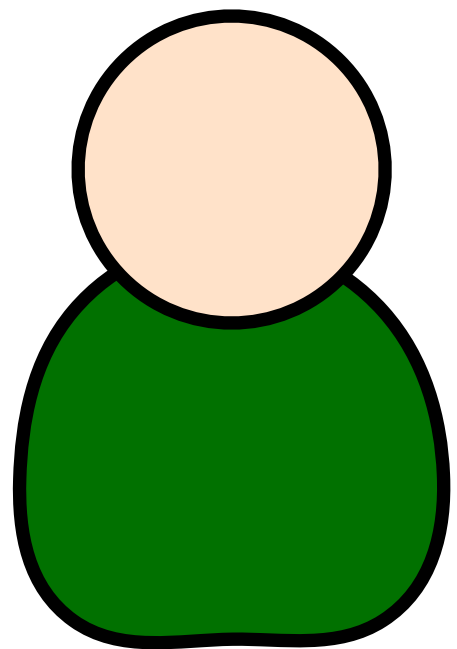
Samia



10

```
typedef struct
{
    string name;
    int votes;
}
candidate;

candidate candidates[4];
```



Plurality

Recursion

Factorial

$$1! = 1$$

$$2! = 2 * 1$$

$$3! = 3 * 2 * 1$$

$$4! = 4 * 3 * 2 * 1$$

Factorial

$$1! = 1$$

$$2! = 2 * 1$$

$$3! = 3 * 2 * 1$$

$$4! = 4 * 3 * 2 * 1$$

Factorial

$$4! = ?$$

Factorial

$$4! = 4 * \underline{3!}$$

Recursive call

Factorial

$$4! = 4 * 3!$$

$$3! = 3 * 2!$$

$$2! = 2 * 1!$$

$$1! = 1$$

Base case

Factorial

$$4! = 4 * 3!$$

$$3! = 3 * 2!$$

$$2! = 2 * 1!$$

$$1! = 1$$



Call stack

Factorial

$$4! = 4 * 3!$$

$$3! = 3 * 2!$$

$$2! = 2 * 1$$

Factorial

$$4! = 4 * 3!$$

$$3! = 3 * 2 * 1$$

Factorial

$$4! = 4 * 3 * 2 * 1$$

Creating a Factorial Function

- In a file called **factorial.c**, implement a function called **factorial** to return the factorial of a given number.
- Call **factorial** from **main** and print the result from **factorial**.

The week ahead

- **Submit Problem Set 3** by Sunday, February 19, 11:59 PM.
- Attend **office hours**.
- Complete <https://cs50.ly/studybuddy> to be paired with a classmate if you'd like!