This is CS50.

cs50.brianyu.me

Week 3

- Searching (Linear, Binary)
- Sorting (Bubble, Selection)
- Big O
- Structs
- Recursion
- Merge Sort

What questions do you have?

Today

Sorting

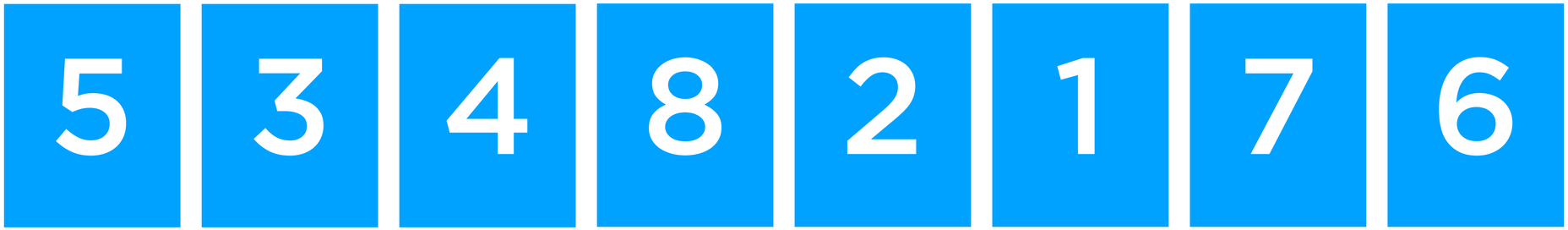
Recursion

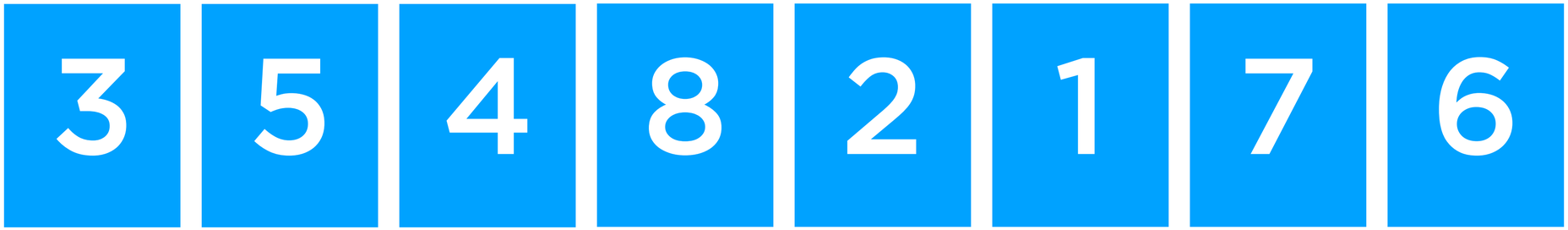
Structs

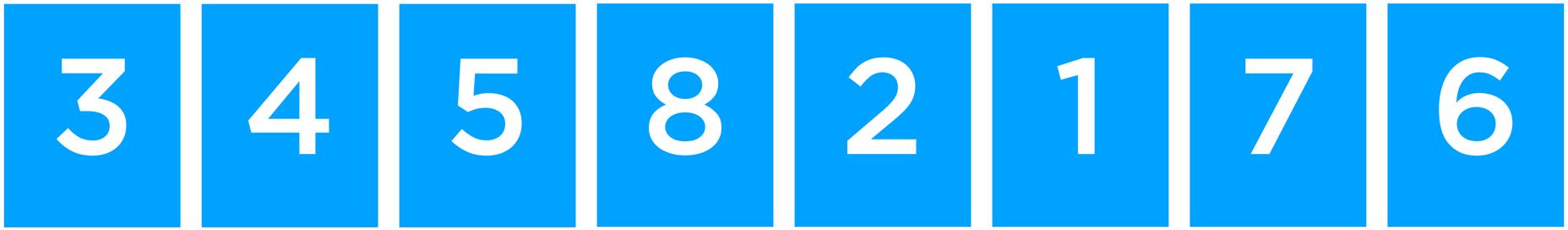
Part One

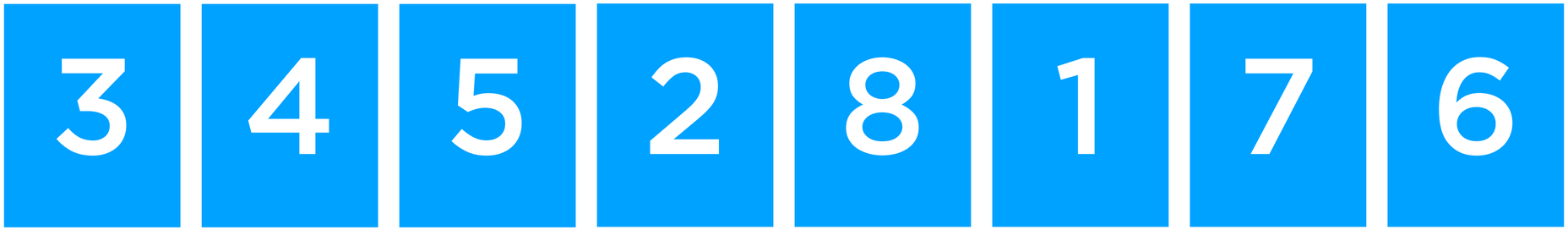
Search and Sort

Bubble Sort


5 3 4 8 2 1 7 6


3 5 4 8 2 1 7 6


3 4 5 8 2 1 7 6


3 4 5 2 8 1 7 6


3 4 5 2 1 8 7 6


3 4 5 2 1 7 8 6


3 4 5 2 1 7 6 8


3
4
5
2
1
7
6
8


3 4 2 5 1 7 6 8


3
4
2
1
5
7
6
8


3
4
2
1
5
6
7
8


3 4 2 1 5 6 7 8


3
2
4
1
5
6
7
8


3
2
1
4
5
6
7
8


3
2
1
4
5
6
7
8


2 3 1 4 5 6 7 8


2 1 3 4 5 6 7 8


2 1 3 4 5 6 7 8


1
2
3
4
5
6
7
8


1
2
3
4
5
6
7
8


1
2
3
4
5
6
7
8

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

```
wget http://cdn.cs50.net/2020/spring/classes/3/sorting.c
```

Exercise

Complete **sorting.c** such that it sorts integers using Bubble Sort.

```
wget http://cdn.cs50.net/2020/spring/classes/3/sorting.c
```

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

Bubble Sort

```
for (int i = 0; i < n - 1; i++)
{
    for (int j = 0; j < n - 1; j++)
    {
        if (values[j] > values[j + 1])
        {
            int temp = values[j];
            values[j] = values[j + 1];
            values[j + 1] = temp;
        }
    }
}
```

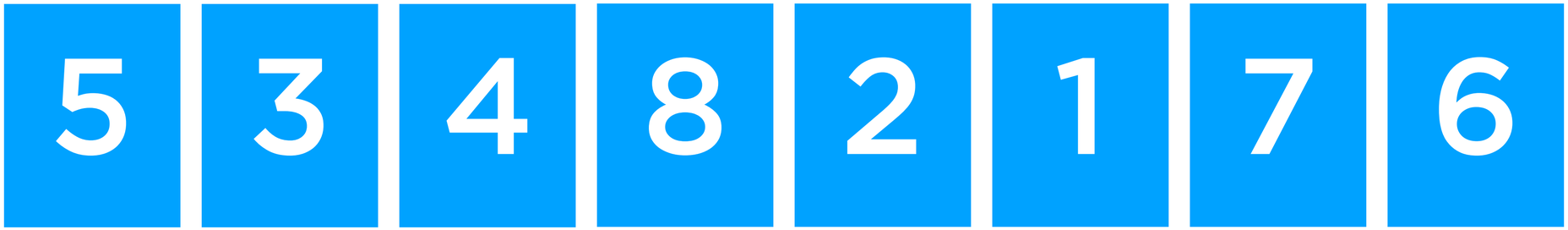
Bubble Sort

```
for (int i = 0; i < n - 1; i++)
{
    for (int j = 0; j < n - 1 - i; j++)
    {
        if (values[j] > values[j + 1])
        {
            int temp = values[j];
            values[j] = values[j + 1];
            values[j + 1] = temp;
        }
    }
}
```

Bubble Sort

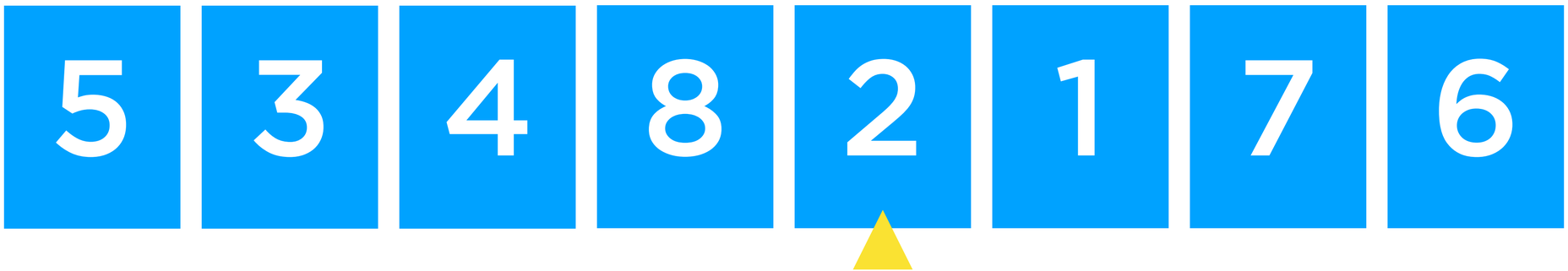
```
for (int i = 0; i < n - 1; i++)
{
    int swaps = false;
    for (int j = 0; j < n - 1 - i; j++)
    {
        if (values[j] > values[j + 1])
        {
            int temp = values[j];
            values[j] = values[j + 1];
            values[j + 1] = temp;
            swaps = true;
        }
    }
    if (swaps == false)
        break;
}
```

Selection Sort

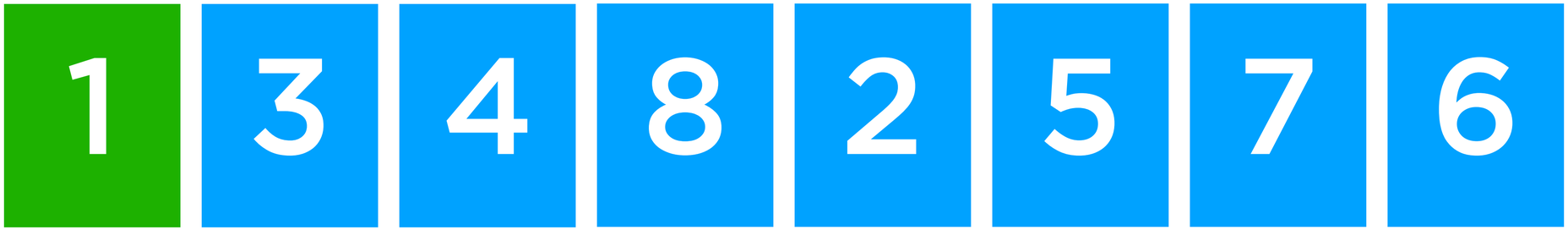

5 3 4 8 2 1 7 6


5 3 4 8 2 1 7 6

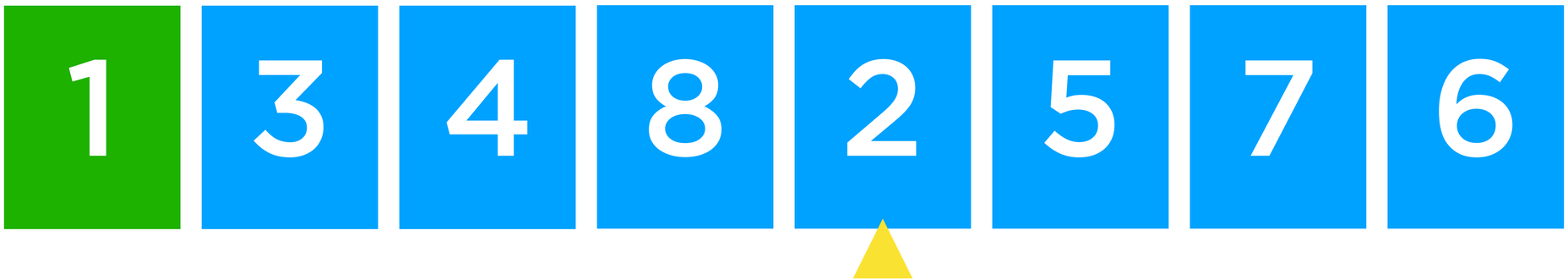

5 3 4 8 2 1 7 6

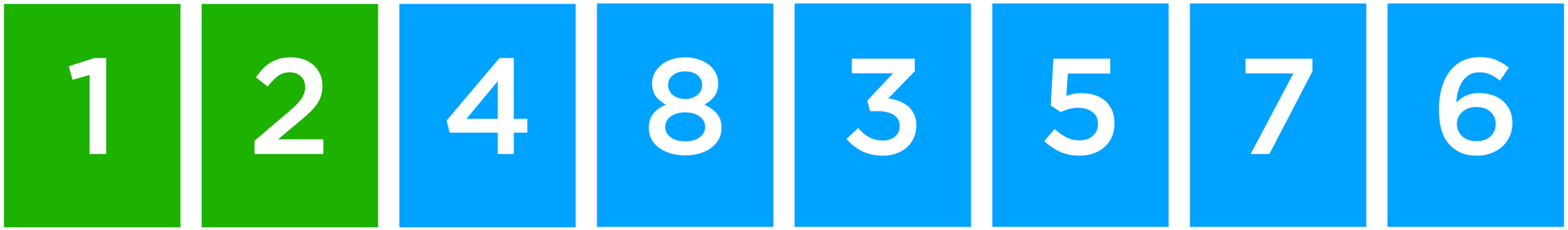

5 3 4 8 2 1 7 6


5 3 4 8 2 1 7 6

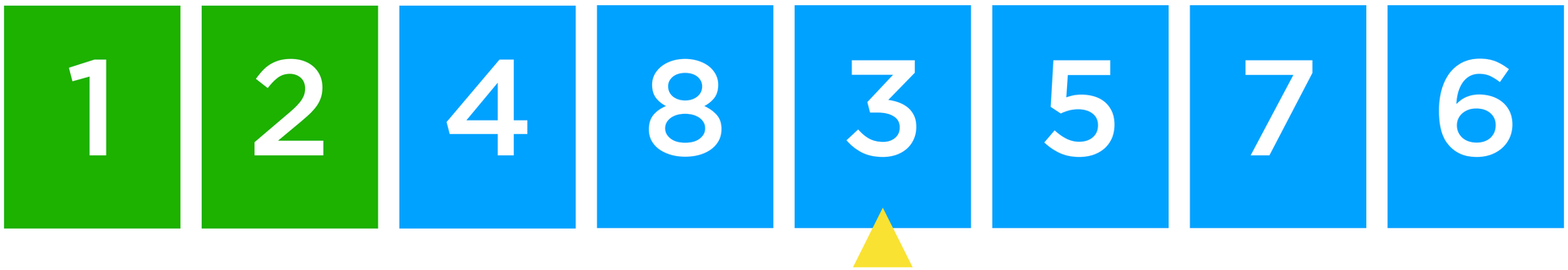

1
3
4
8
2
5
7
6

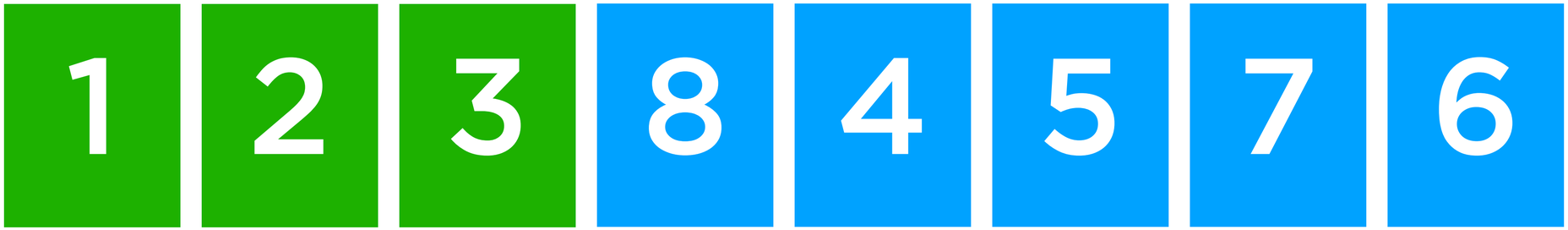

1
3
4
8
2
5
7
6

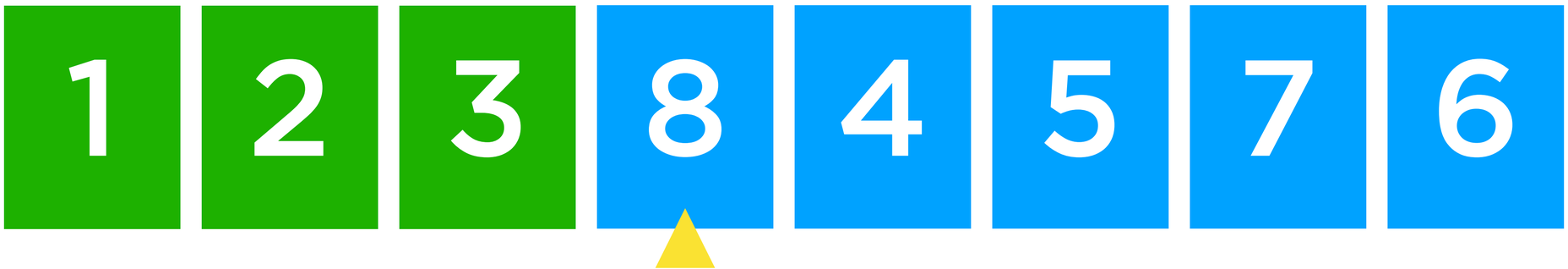

1
3
4
8
2
5
7
6

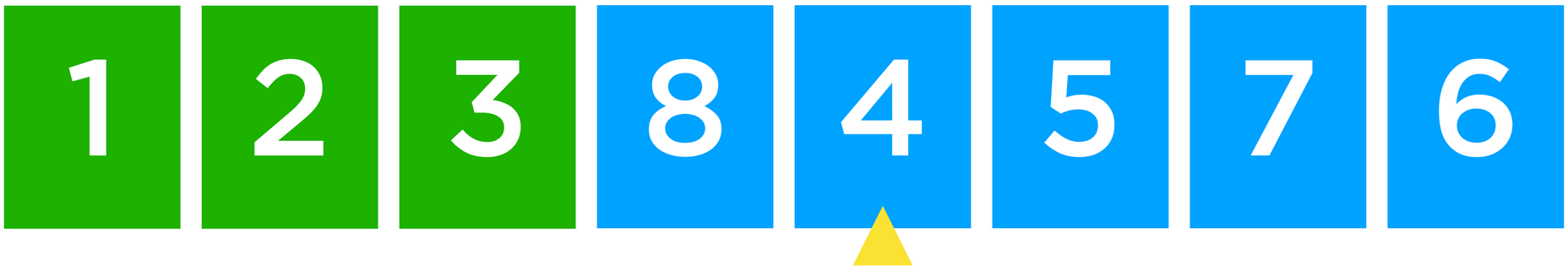

1
2
4
8
3
5
7
6


1
2
4
8
3
5
7
6


1
2
4
8
3
5
7
6


1 2 3 8 4 5 7 6


1
2
3
8
4
5
7
6


1
2
3
8
4
5
7
6


1 2 3 4 8 5 7 6


1
2
3
4
8
5
7
6


1
2
3
4
8
5
7
6


1 2 3 4 5 8 7 6


1
2
3
4
5
8
7
6


1 2 3 4 5 8 7 6


1 2 3 4 5 8 7 6


1
2
3
4
5
6
7
8


1
2
3
4
5
6
7
8


1
2
3
4
5
6
7
8


1
2
3
4
5
6
7
8

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

Exercise

Complete `sorting.c` such that it sorts integers using Selection Sort.

```
wget http://cdn.cs50.net/2020/spring/classes/3/sorting.c
```

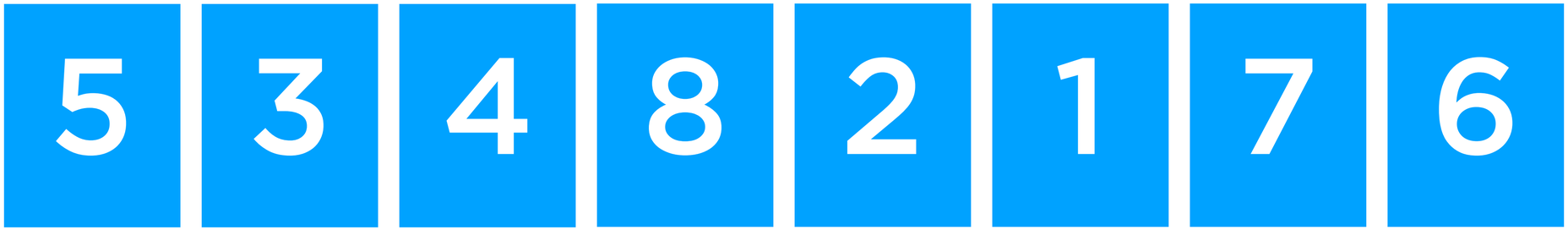
```
For i from 0 to n-1
    Find smallest item between i'th item and last item
    Swap smallest item with i'th item
```

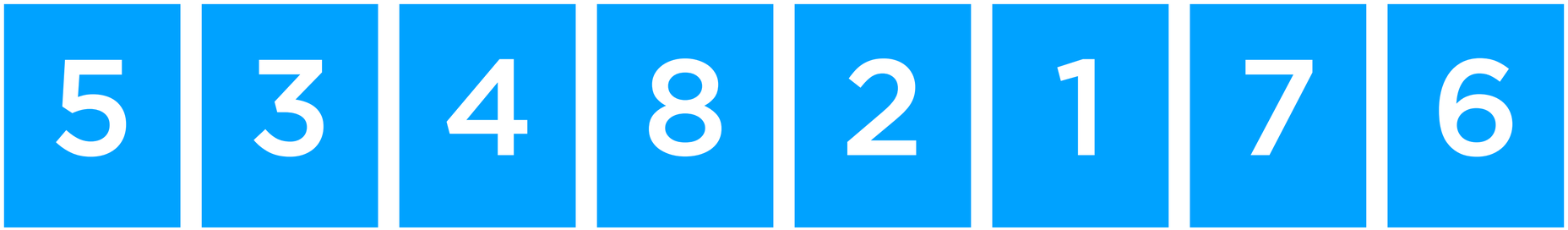
Selection Sort

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

```
for (int i = 0; i < n - 1; i++)
{
    int min_index = i;
    for (int j = i + 1; j < n; j++)
    {
        if (values[j] < values[min_index])
        {
            min_index = j;
        }
    }
    int temp = values[i];
    values[i] = values[min_index];
    values[min_index] = temp;
}
```

Mergesort


5 3 4 8 2 1 7 6


5 3 4 8 2 1 7 6


5 3 4 8
2 1 7 6


5
3
4
8


2
1
7
6

2 1 7 6

5 3

4 8

2 1 7 6

5 3

4 8

2176

48

53

2176

48

5

3

2176

48

5

3

2176

48

5

3

2176

48

5 3

2176

48

5 3

2 1 7 6

3

4 8

5

2 1 7 6

3 5

4 8

2 1 7 6

3 5

4 8

2 1 7 6

3 5

4 8

2 1 7 6

3 5

4 8

2 1 7 6

3 5

4

8

2 1 7 6

3 5

4

8

2 1 7 6

3 5

4

8

2 1 7 6

3 5

4 8

2 1 7 6

3 5

4 8

2 1 7 6

3 5

4

8

2 1 7 6

3 5 4 8

2176

35 48

2 1 7 6

3 5

4 8


3


2
1
7
6


5
4
8


3


2
1
7
6


5
4
8


3
4


2
1
7
6


5
8


3
4


2
1
7
6


5
8


3
4
5


2
1
7
6


8

3 4 5 8

2 1 7 6


3
4
5
8


2
1
7
6

3 4 5 8

2 1 7 6


3
4
5
8


2
1


7
6


3
4
5
8


2
1


7
6

3 4 5 8

7 6

2 1


3
4
5
8


7
6


2
1

3 4 5 8

7 6

2

1

3 4 5 8

7 6

2

1

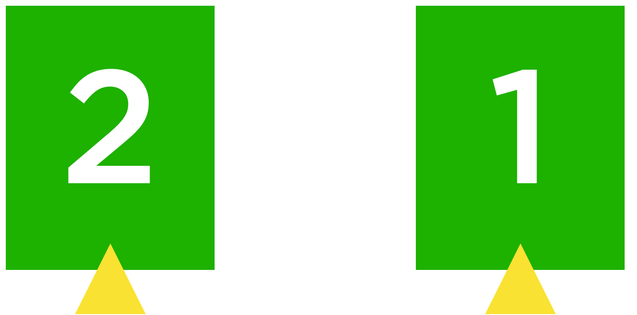
3 4 5 8

7 6

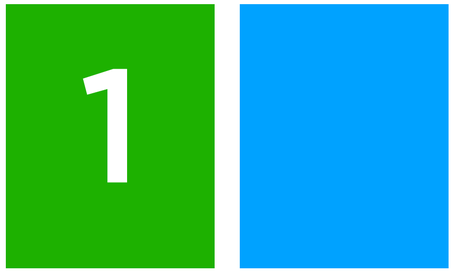
2 1


3
4
5
8


7
6


2
1


3
4
5
8


1


7
6


2

3 4 5 8

1 2

7 6


3
4
5
8


1
2


7
6


3
4
5
8


1
2


7
6


3
4
5
8


1
2


7


6


3
4
5
8


1
2


7


6


3
4
5
8


1
2


7


6


3
4
5
8


1
2


7


6

3 4 5 8

1 2

7

6


3
4
5
8


1
2


7


6


3
4
5
8


1
2


6

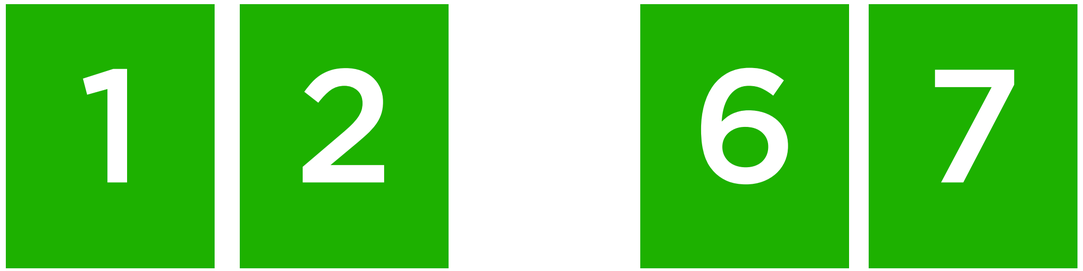

7

3 4 5 8

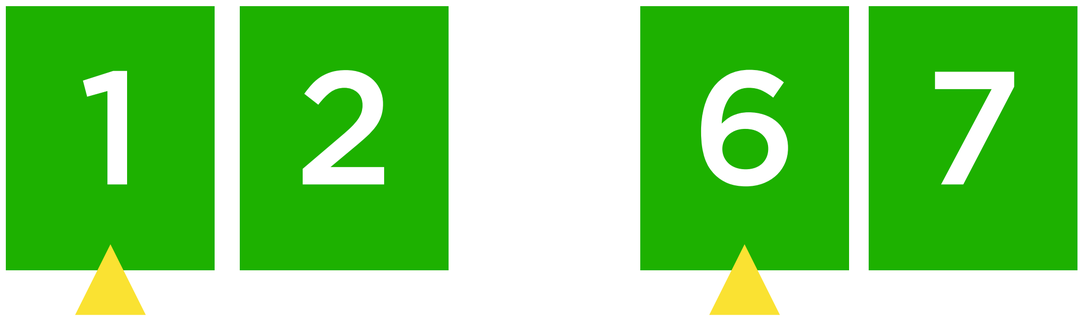
1 2

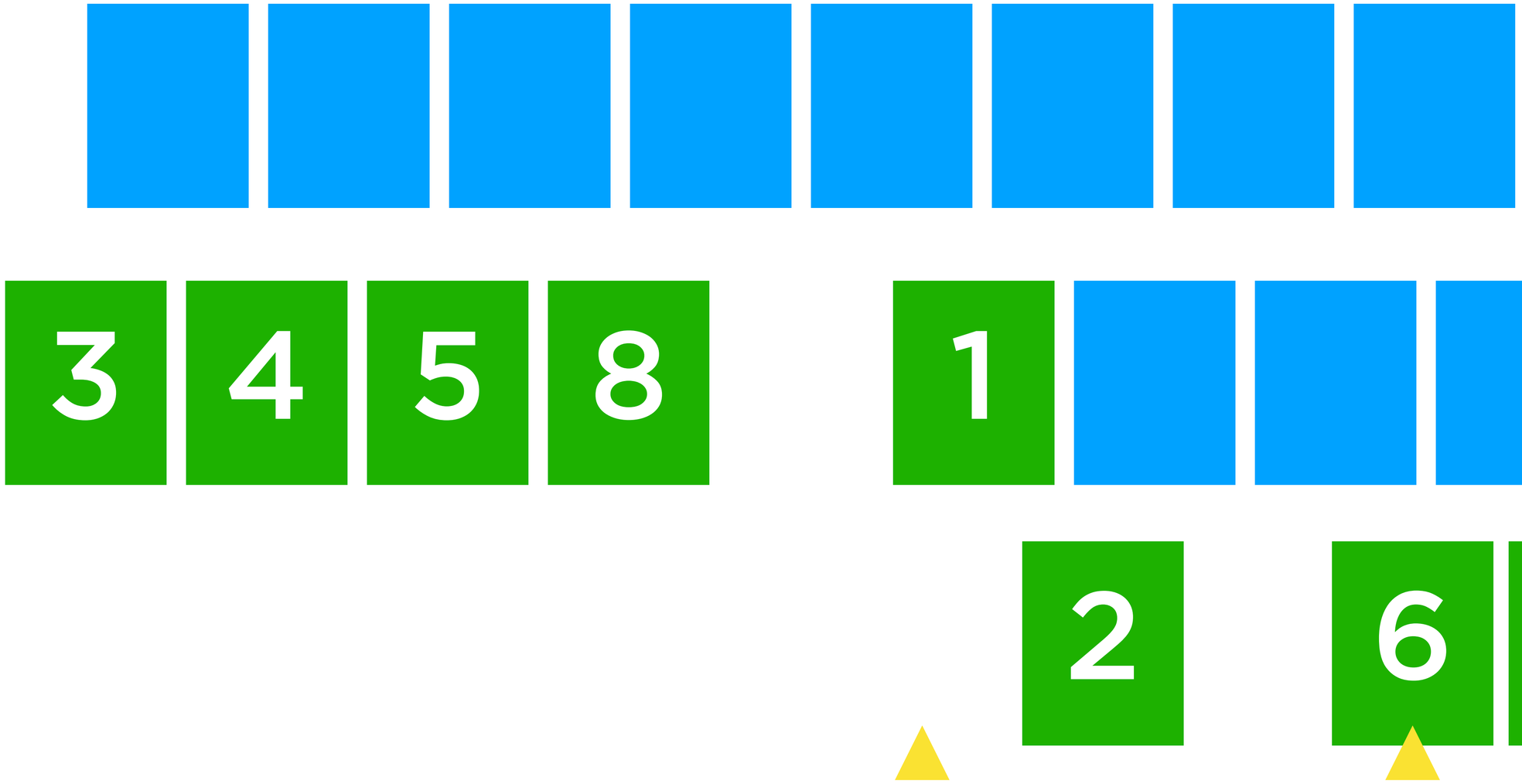
6 7


3
4
5
8

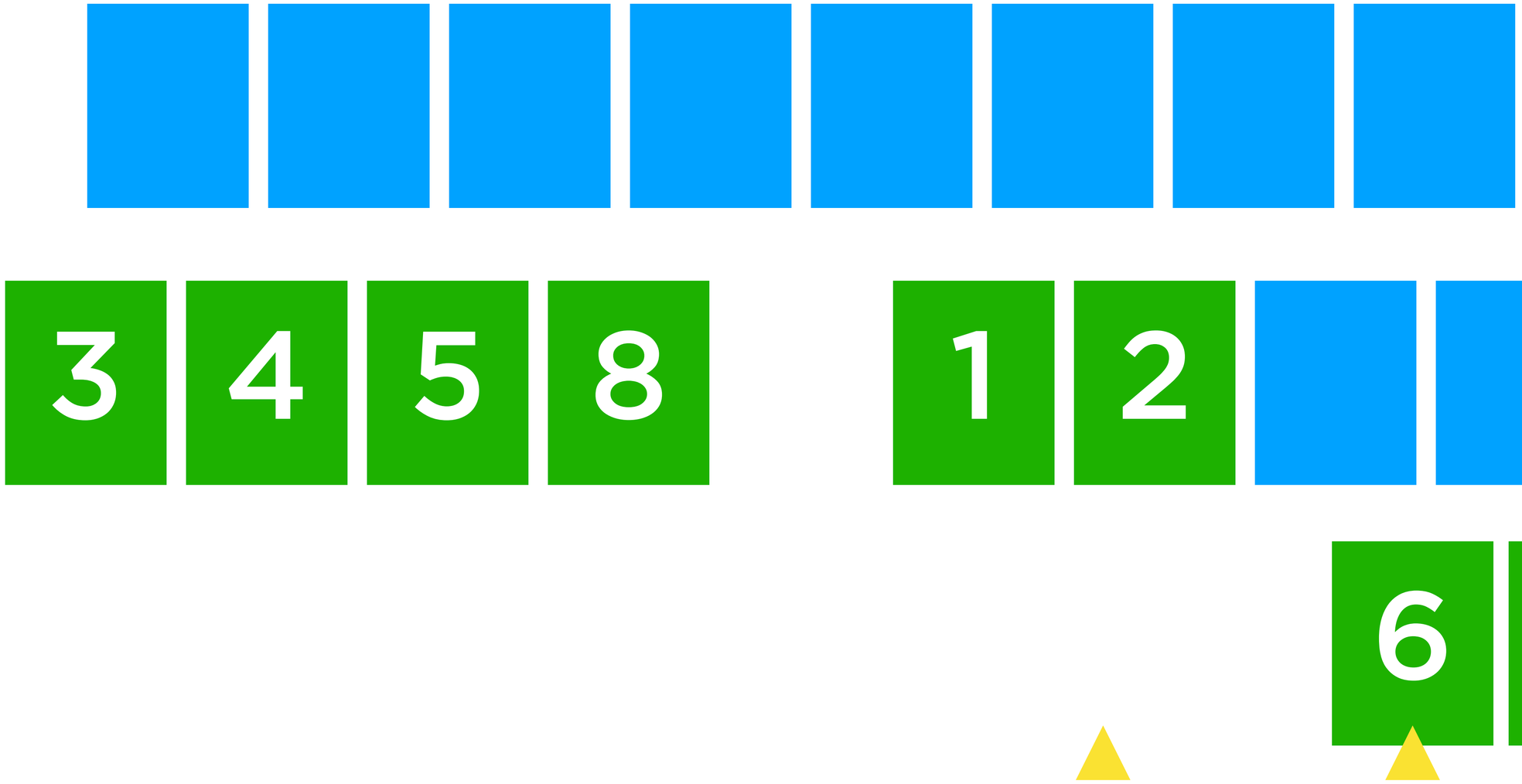

1
2
6
7


3
4
5
8


1
2
6
7


3
4
5
8
1
2
6
7

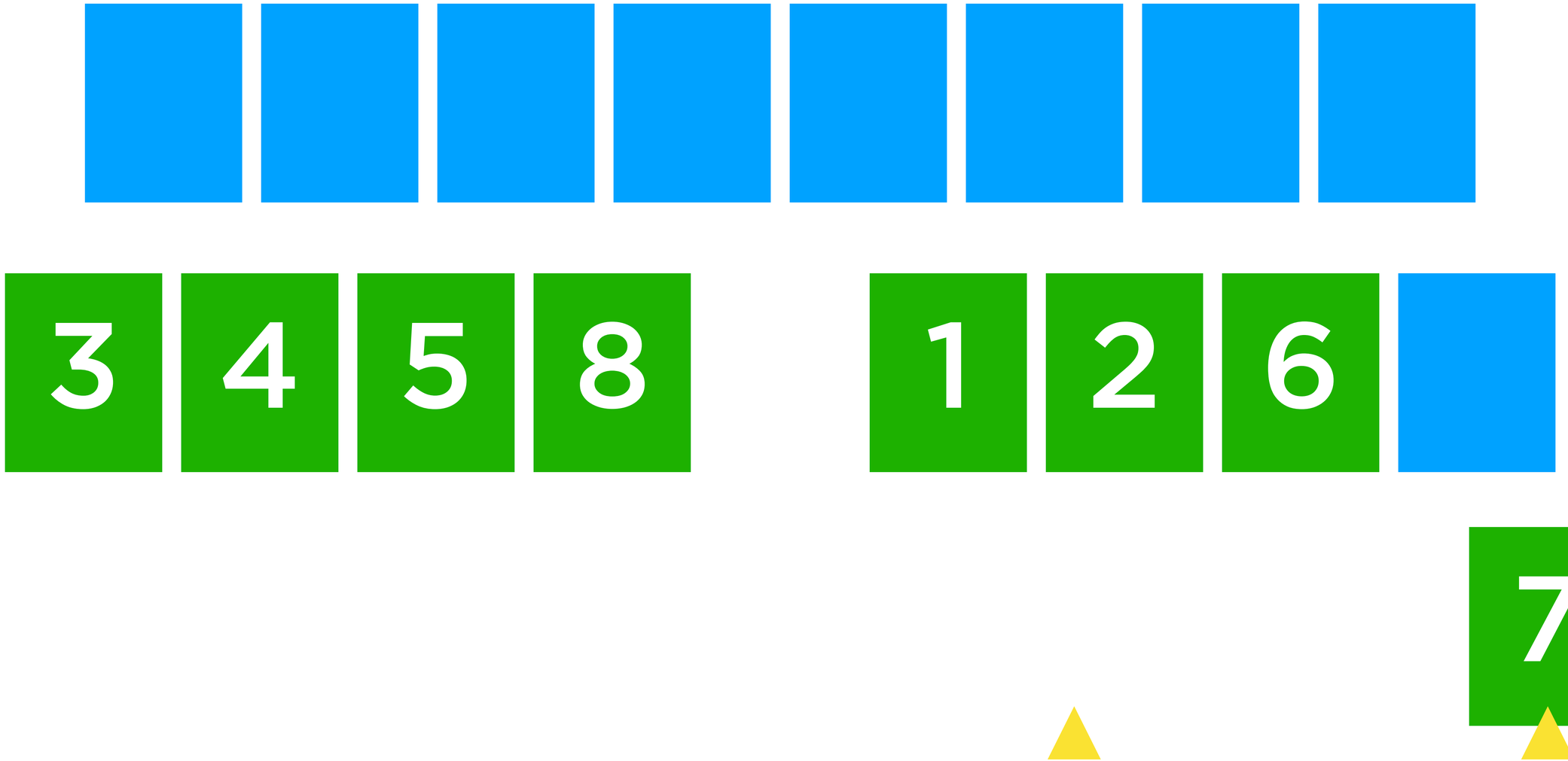

3
4
5
8
1
2
6
7


3
4
5
8
1
2
6
7


3
4
5
8


1
2
6


7

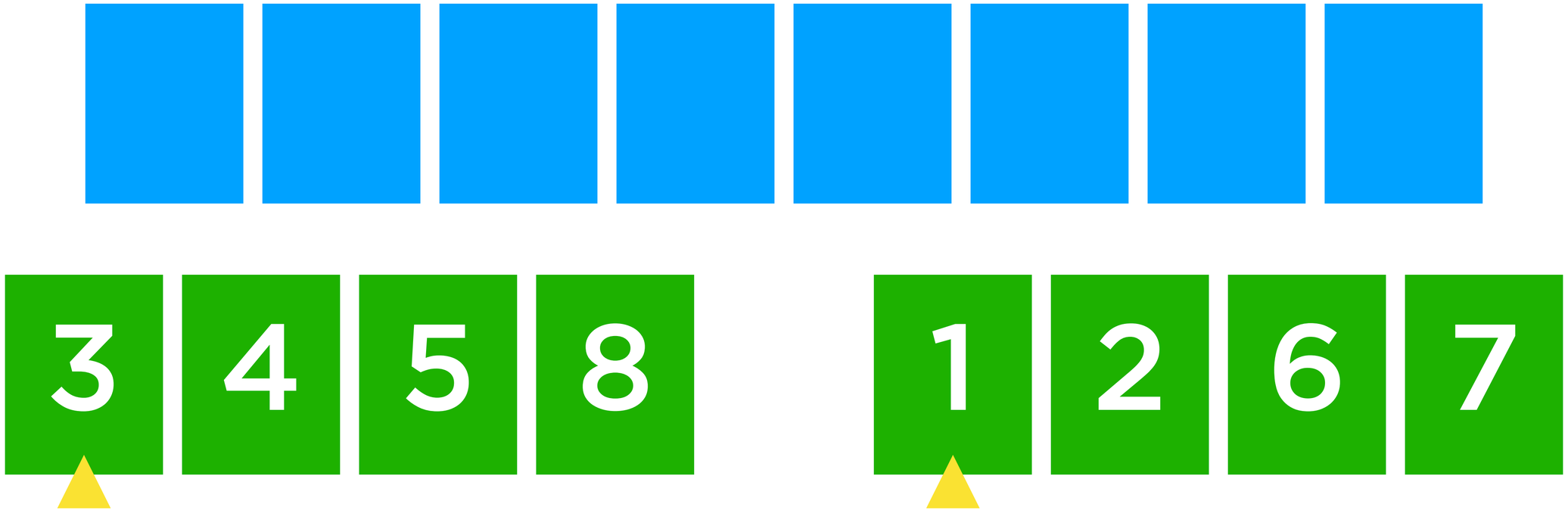

3
4
5
8
1
2
6
7

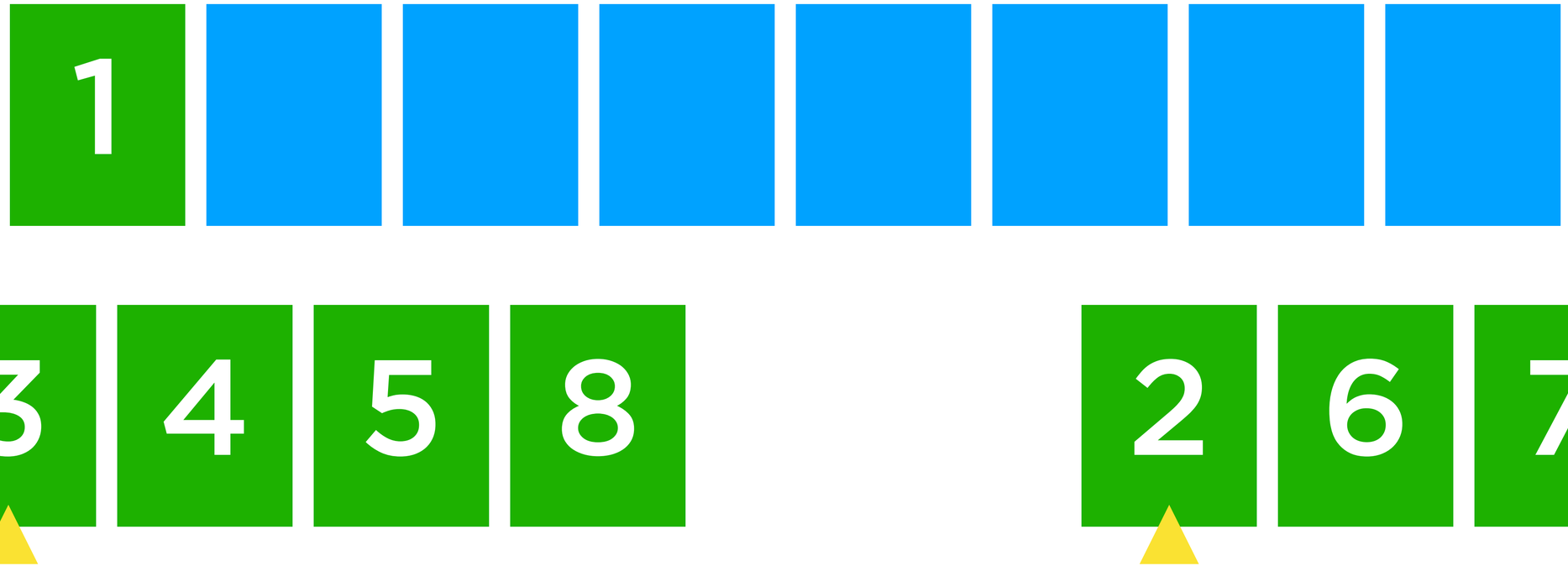
3 4 5 8

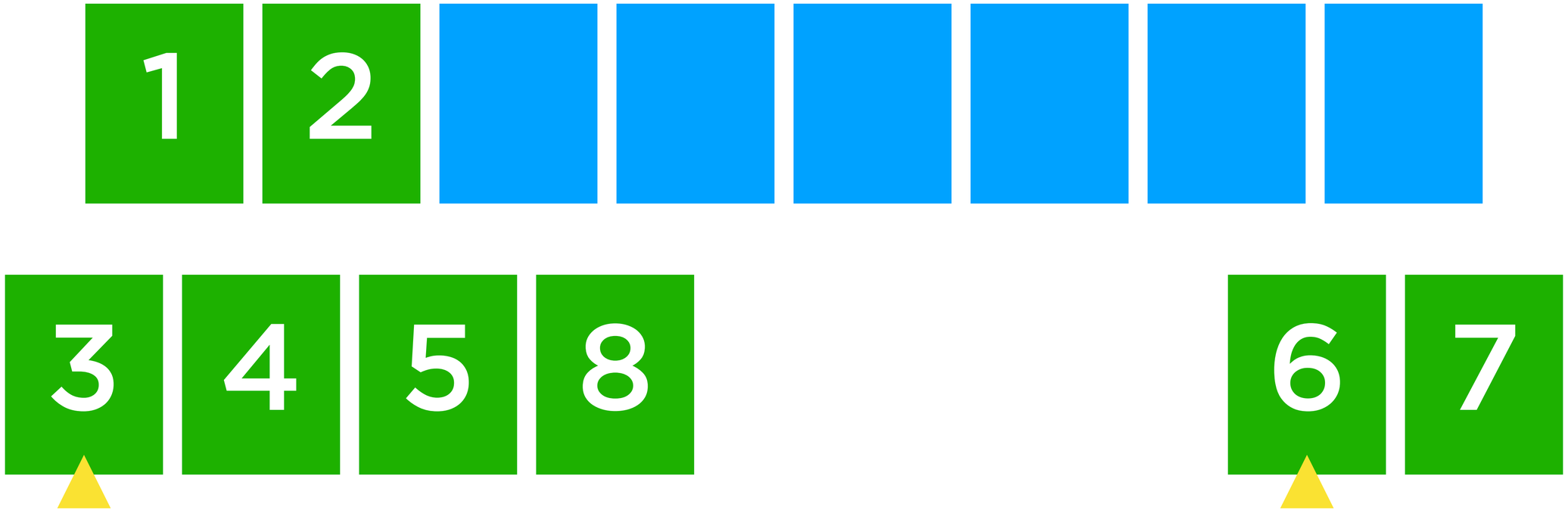
1 2 6 7

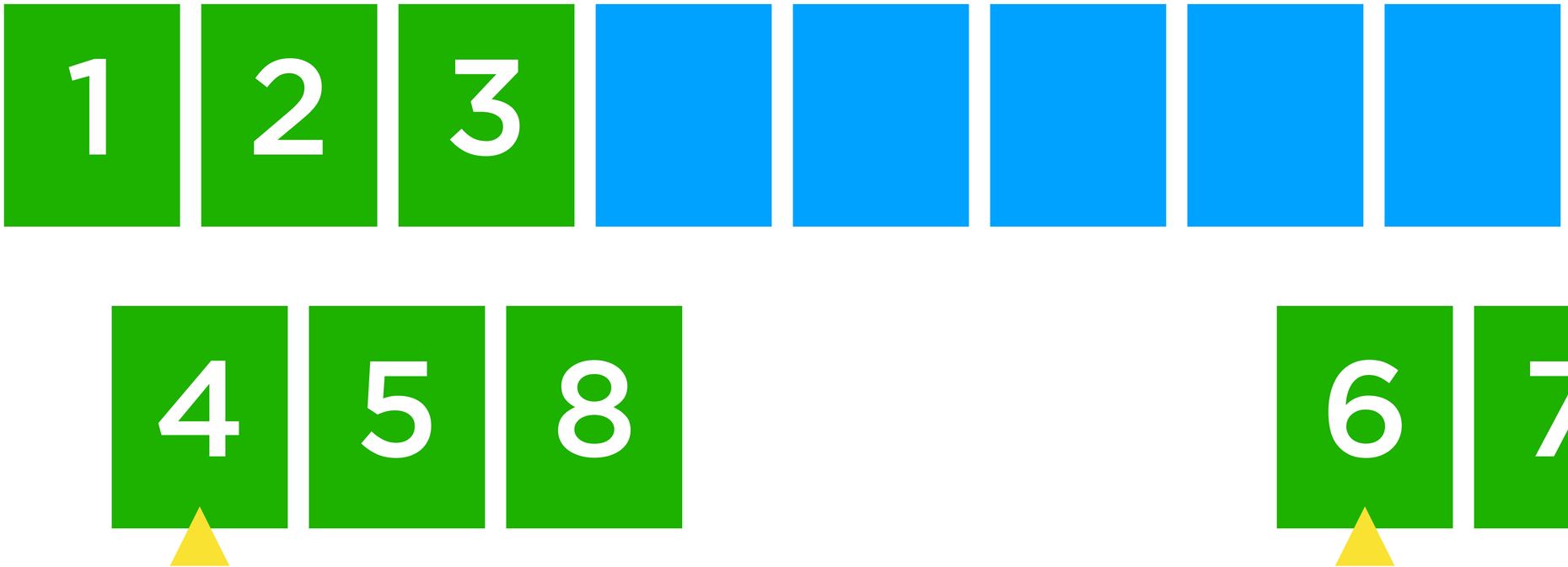
3 4 5 8

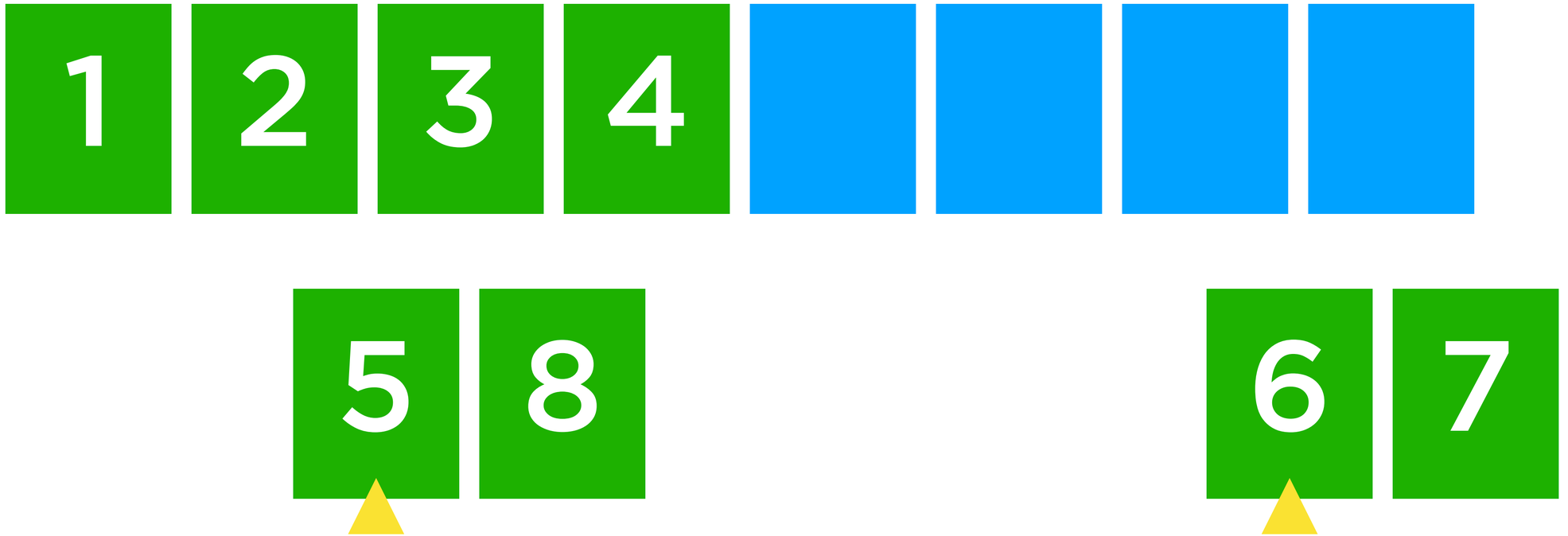
1 2 6 7

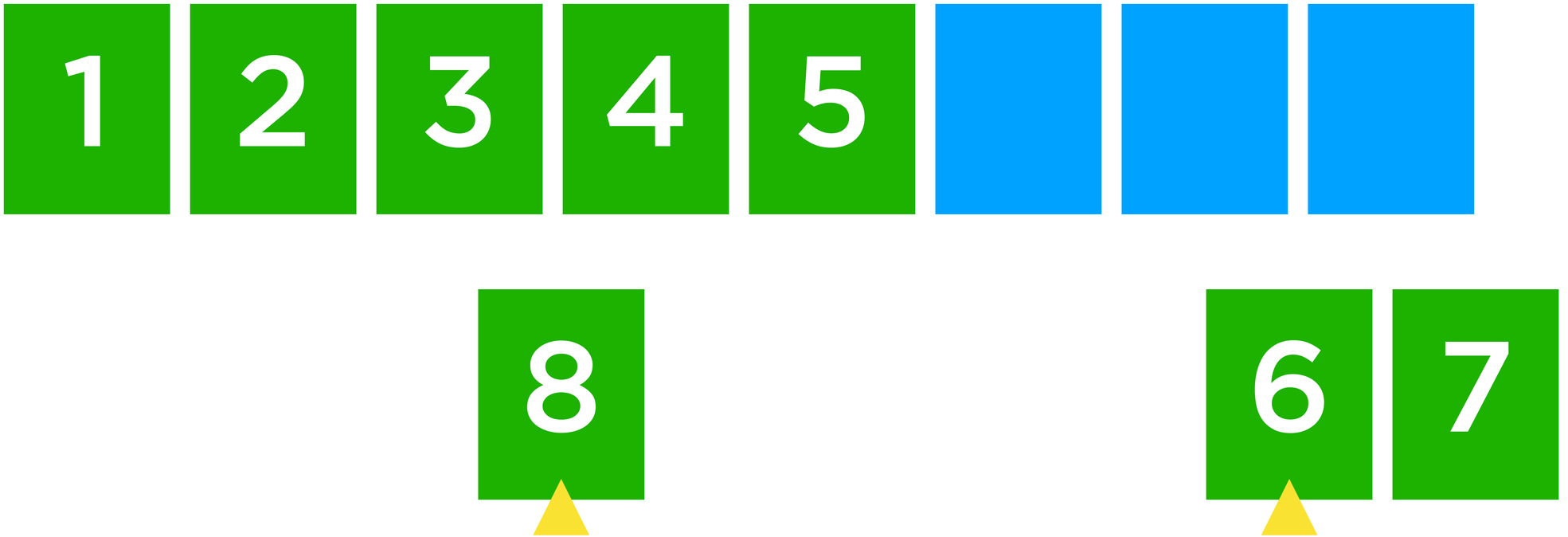

3 4 5 8
1 2 6 7

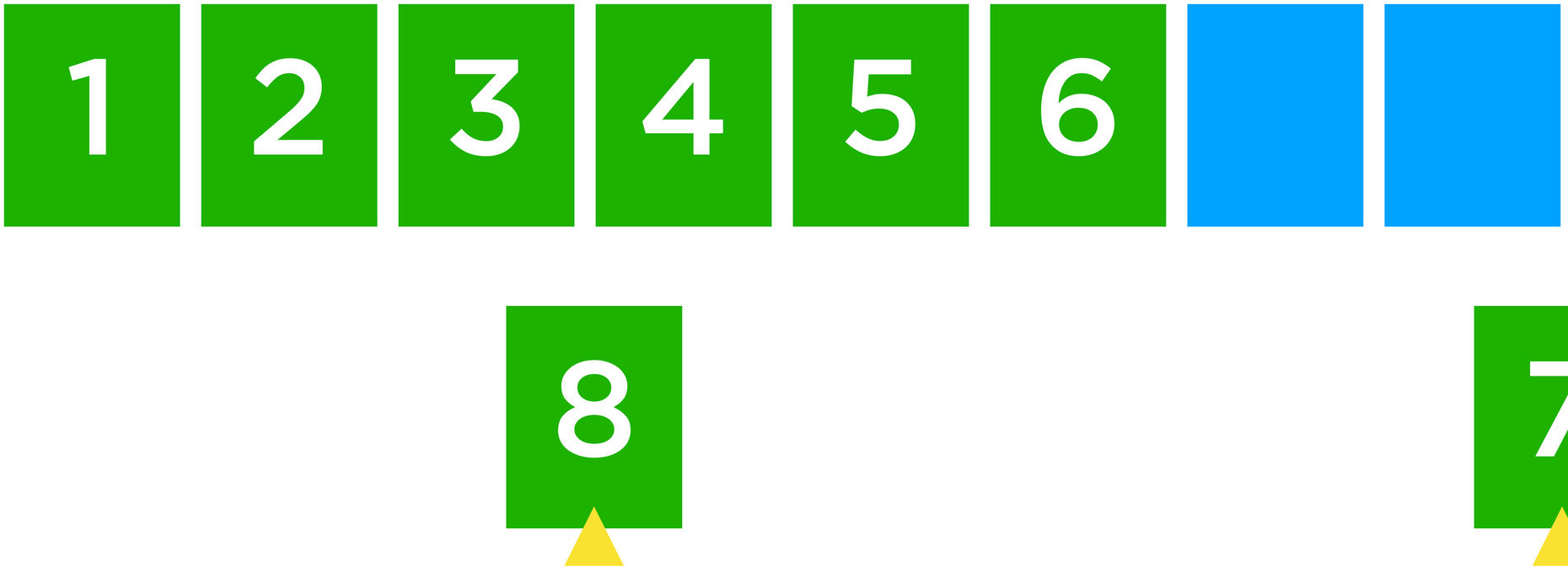

1
3
4
5
8
2
6
7

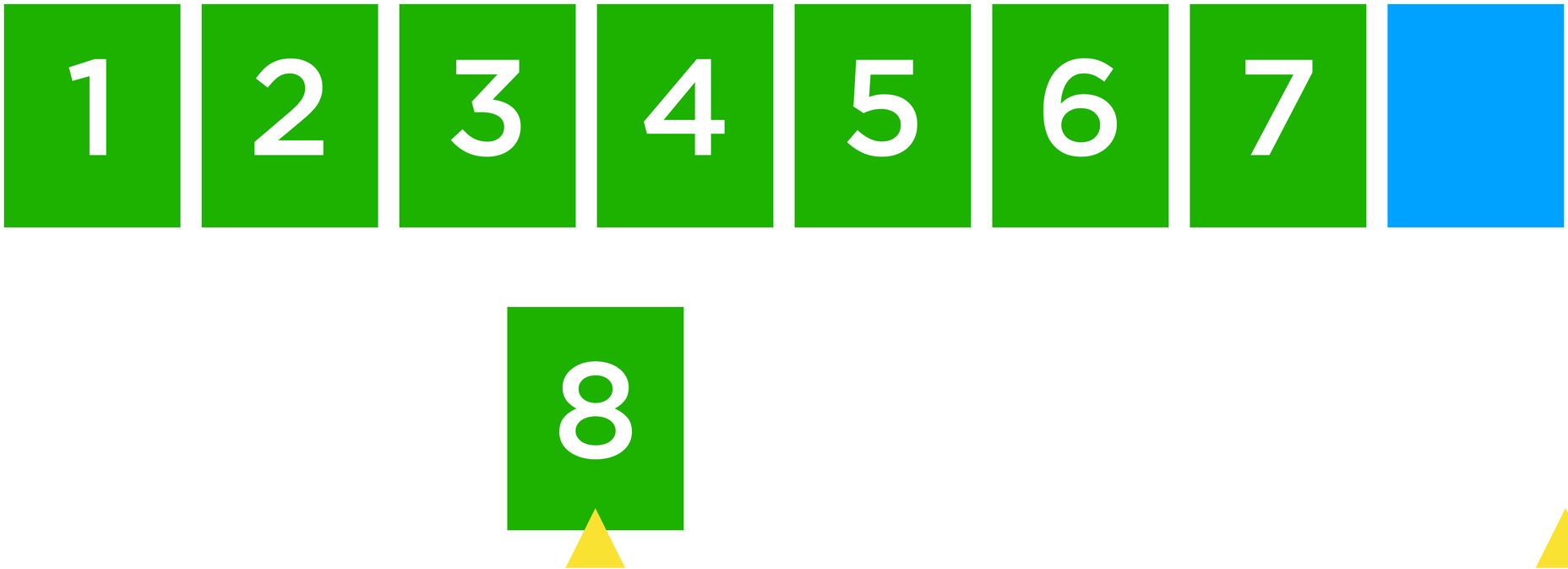

1
2
3
4
5
8
6
7


1
2
3
4
5
8
6
7


1
2
3
4
5
8
6
7


1
2
3
4
5
8
6
7


1
2
3
4
5
6
8
7


1
2
3
4
5
6
7
8


1
2
3
4
5
6
7
8


1
2
3
4
5
6
7
8


1
2
3
4
5
6
7
8

Part Two

Recursion

```
void countdown(int n)
{
    if (n == 0)
        return;
    printf("%i\n", n);
    countdown(n - 1);
}
```

```
void countdown(int n)
{
    if (n == 0)
        return;
    printf("%i\n", n);
    countdown(n - 1);
}
```

```
void countdown(int n)
{
    if (n == 0)
        return;
    printf("%i\n", n);
    countdown(n - 1);
}
```

base case

```c
void countdown(int n)
{
    if (n == 0)
        return;
    printf("%i\n", n);
    countdown(n - 1);
}
```

```
void countdown(int n)
{
    if (n == 0)
        return;
    printf("%i\n", n);
    countdown(n - 1);
}
```

recursive call

```
void countdown(int n)
{
    if (n == 0)
        return;
    printf("%i\n", n);
    countdown(n - 1);
}
```

Exercise

Write a program **`factorial.c`** that calculates the factorial of a number.

e.g. Factorial of 5 is 5 * 4 * 3 * 2 * 1 = 120

```
$ ./factorial
Number: 5
Factorial is 120.
```

Exercise

Write a program **`fib.c`** that prints the nth Fibonacci number.

0th is 0, 1st is 1, all others are sum of two previous.

```
$ ./fib
Number: 5
3
```

Part Three

Structs

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

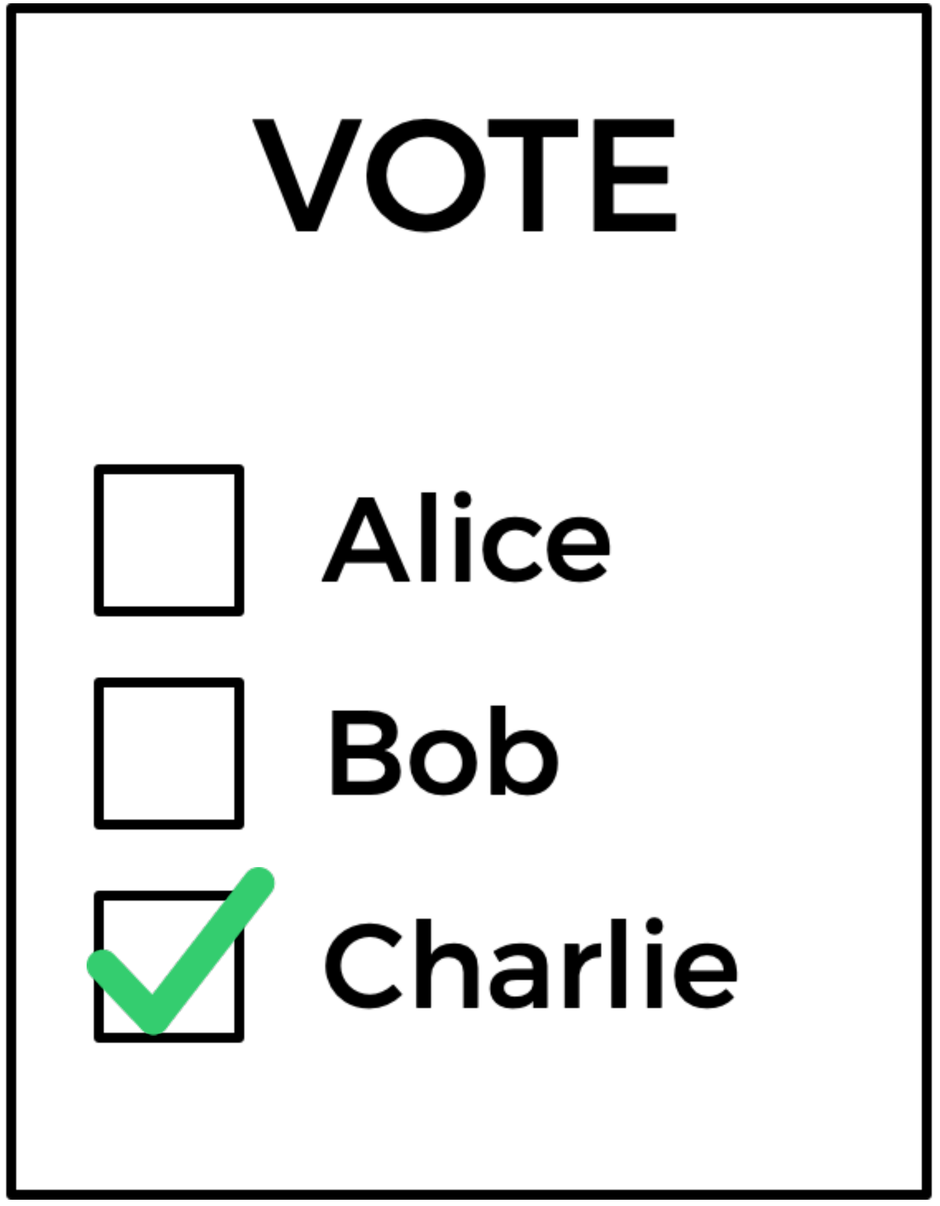
```
typedef struct
{
    string name;
    string number;
}
person;
```

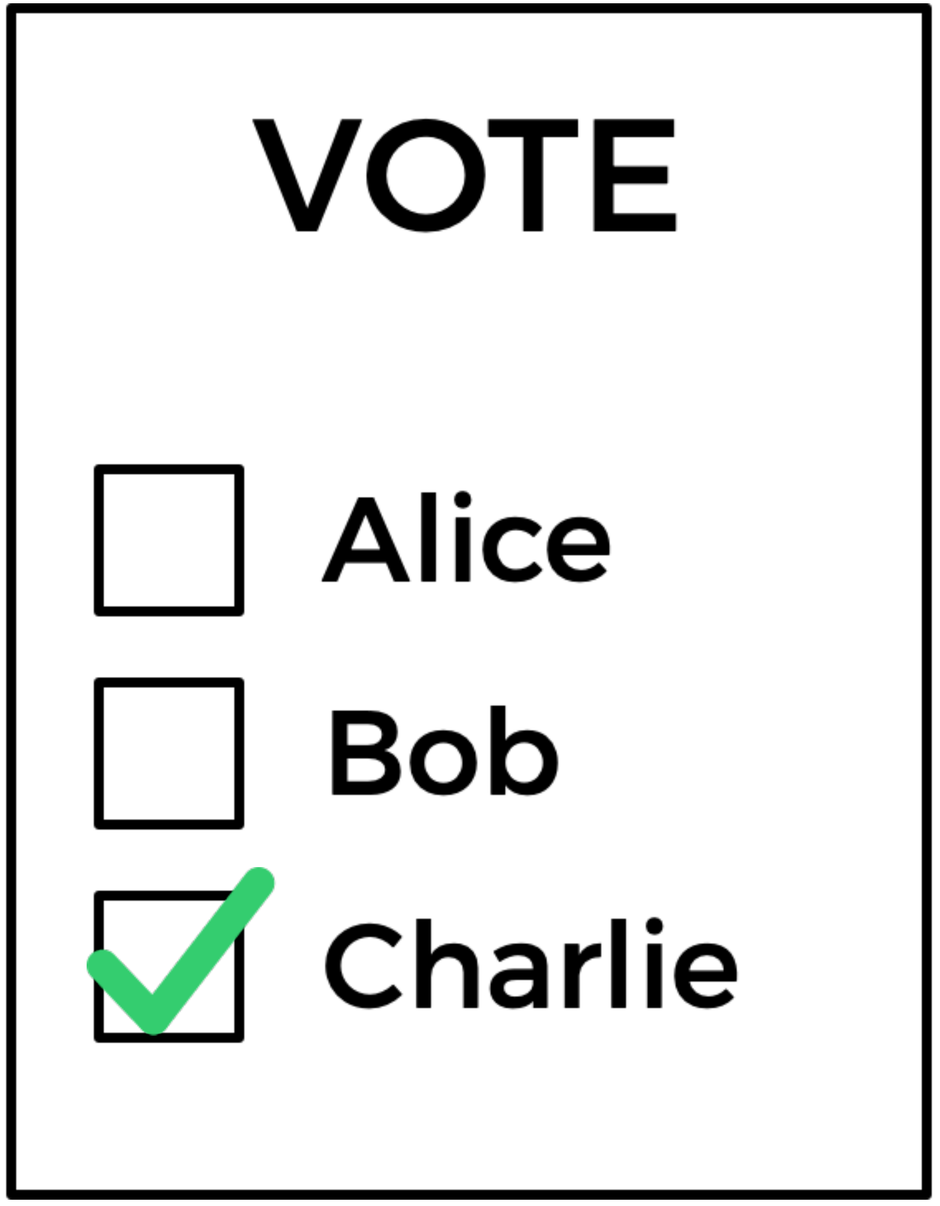
```
person p;

p.name = "Emma";

p.number = "555-0100";
```

Problem Set 3

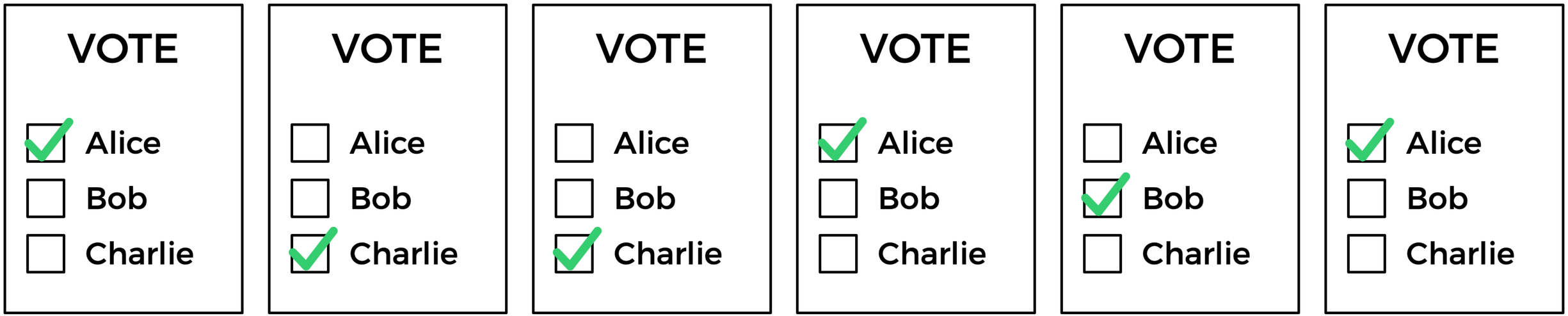
Problem Set 3

- Plurality
- One of:
    - Runoff
    - Tideman

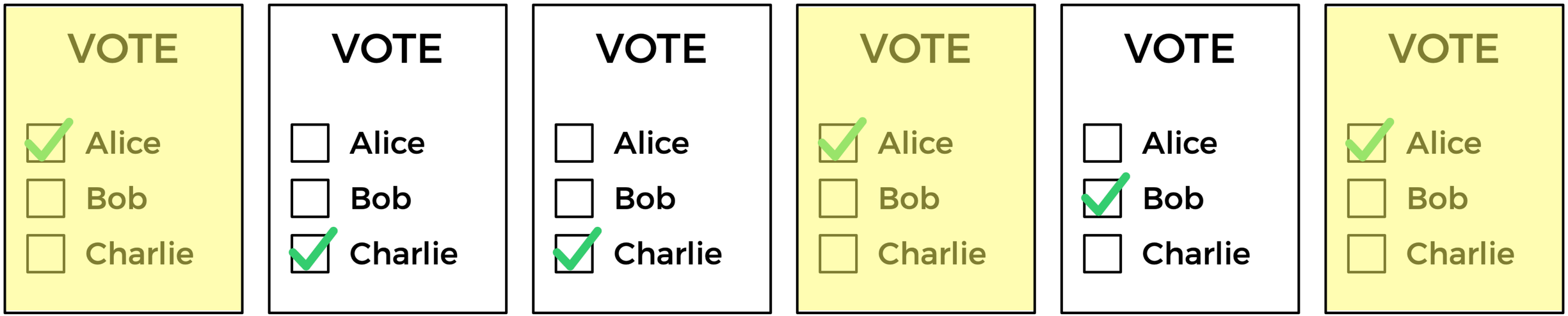

VOTE
Alice
Bob
Charlie


VOTE
Alice
Bob
Charlie

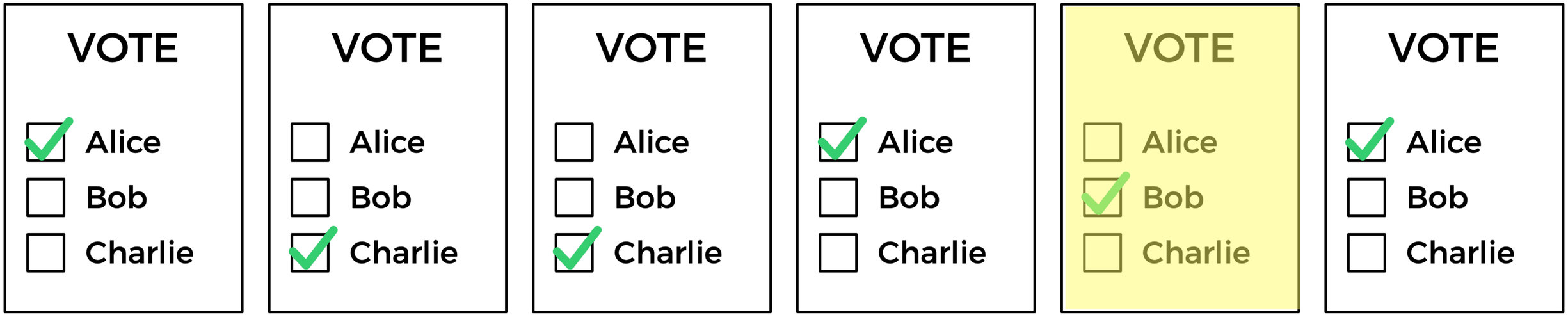
name	Alice	Bob	Charlie
votes			



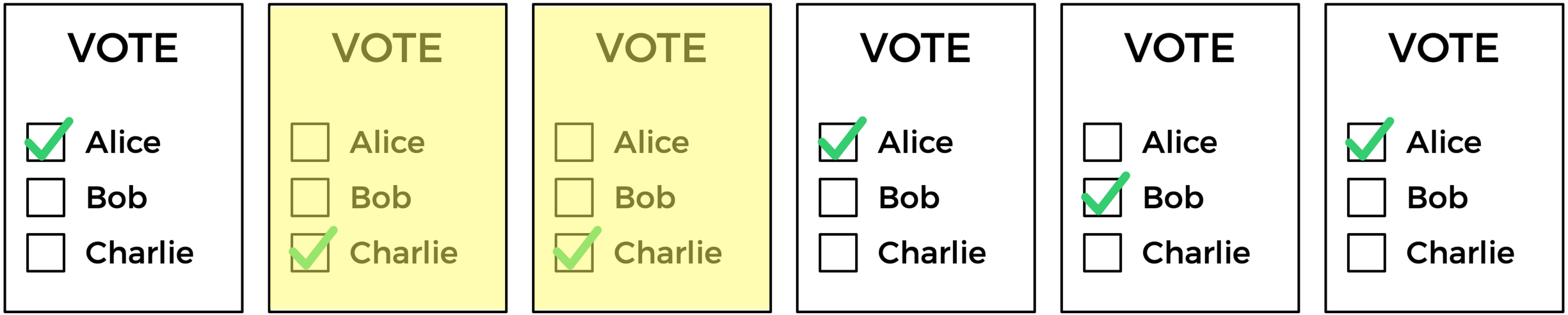
name	Alice	Bob	Charlie
votes	3		



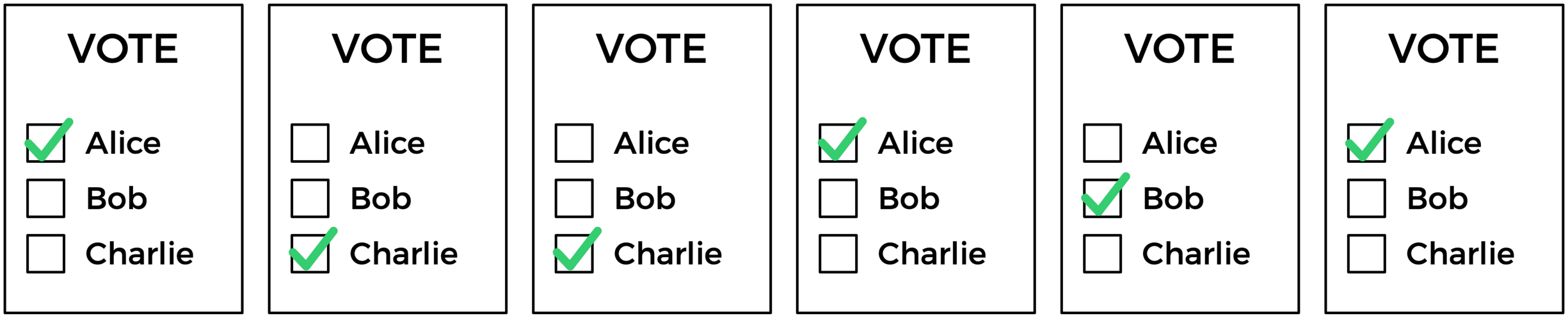
name	Alice	Bob	Charlie
votes	3	1	



name	Alice	Bob	Charlie
votes	3	1	2



name	Alice	Bob	Charlie
votes	3	1	2



Plurality Vote

- Every voter chooses one candidate.
- Whichever candidate has the most votes wins.

```
$ ./plurality Alice Bob Charlie
Number of voters: 4
Vote: Alice
Vote: Bob
Vote: Charlie
Vote: Charlie
Charlie
```

name	Alice	Bob	Charlie
votes	3	1	2

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

candidate candidates[MAX];
```

name	Alice
votes	3

name	Alice	Bob	Charlie
votes	3	1	2

```
candidate candidates[MAX];
```

name	Alice	Bob	Charlie
votes	3	1	2

candidates[0]

name	Alice	Bob	Charlie
votes	3	1	2

candidates[1]

name	Alice	Bob	Charlie
votes	3	1	2

candidates[2]

name	Alice	Bob	Charlie
votes	3	1	2

candidates[2].name

name	Alice	Bob	Charlie
votes	3	1	2

```
candidates[2].votes
```

Ranked-Preference Voting

VOTE

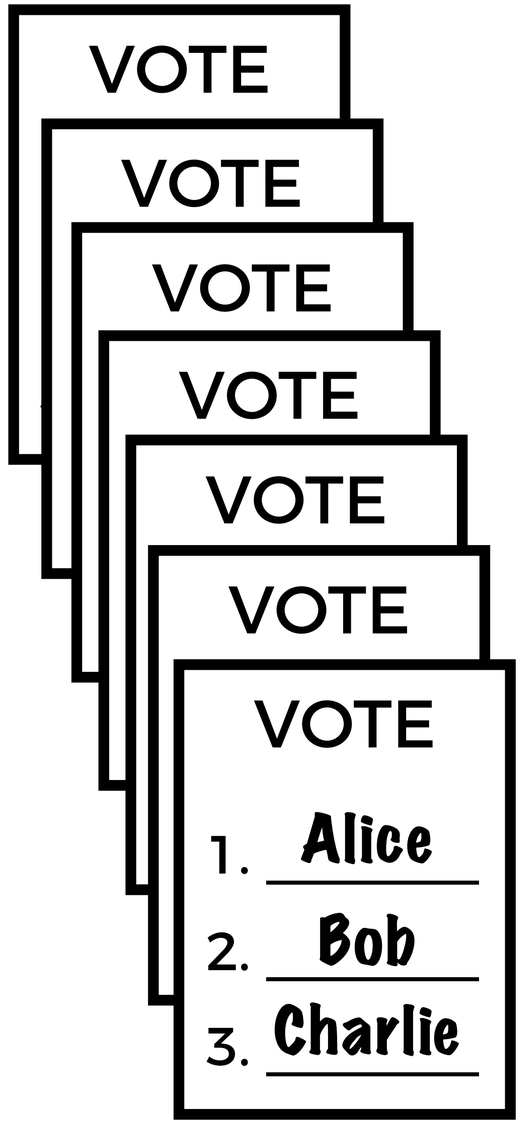
1. Alice
2. Charlie
3. Bob

Runoff Vote

- Every voter ranks their preferences.
- If a candidate has a majority (more than half) of the votes, they are the winner.
- Otherwise, eliminate the candidate with the fewest votes and re-run the election without them.

7 votes.

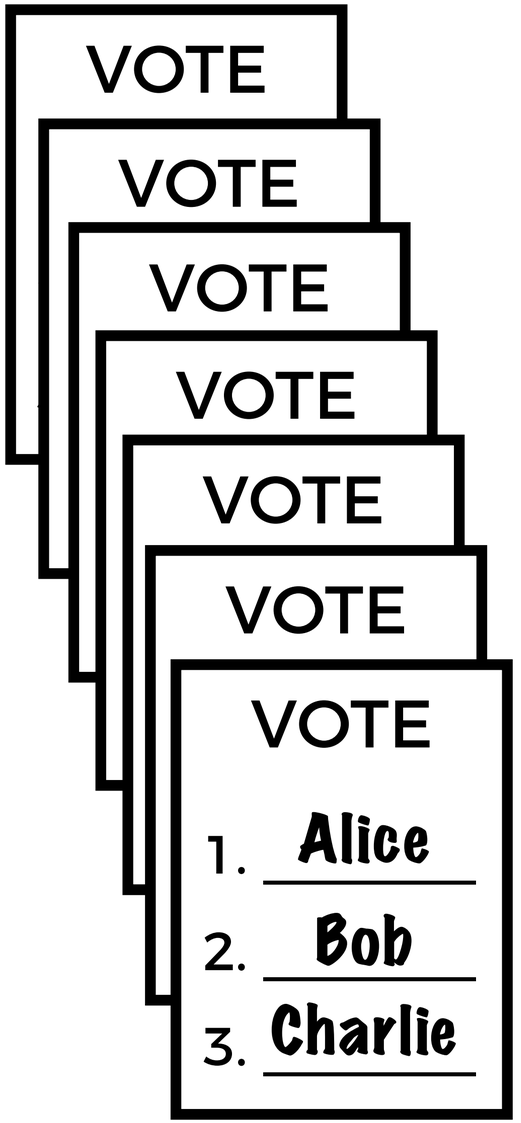
4 votes to win.



Alice

Bob

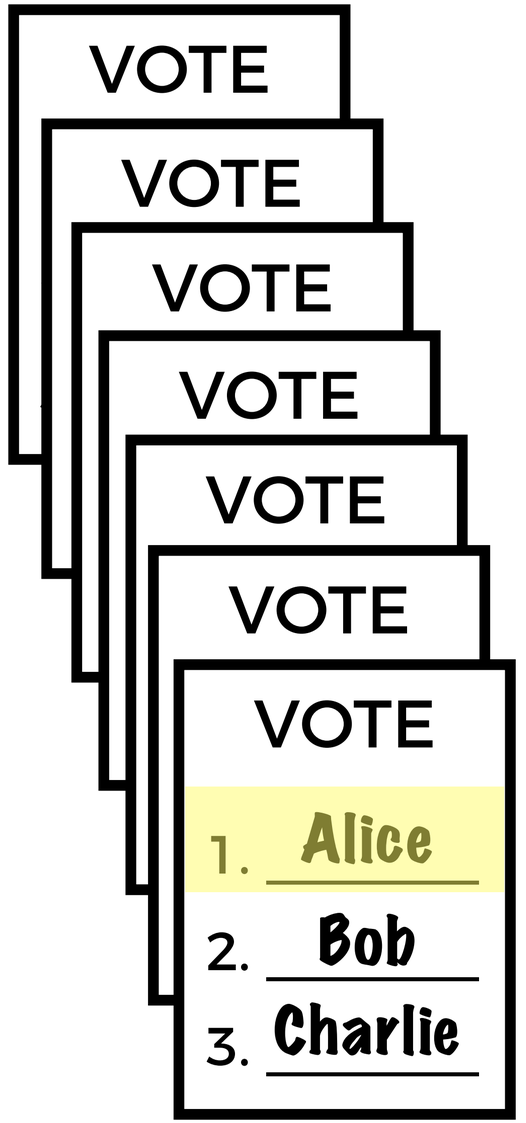
Charlie



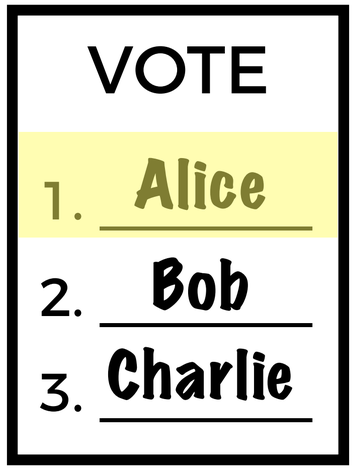
Alice

Bob

Charlie

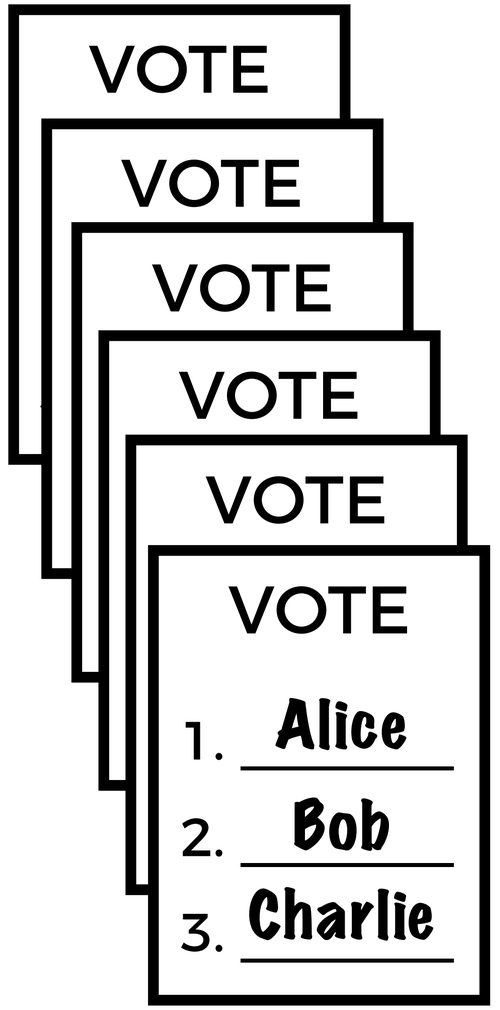


Alice

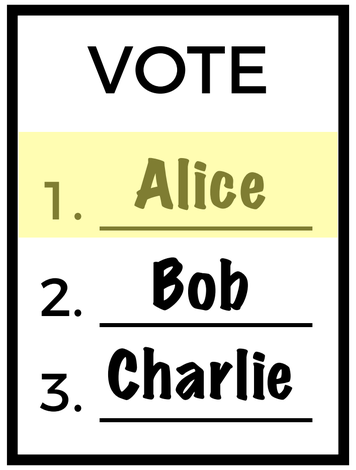


Bob

Charlie

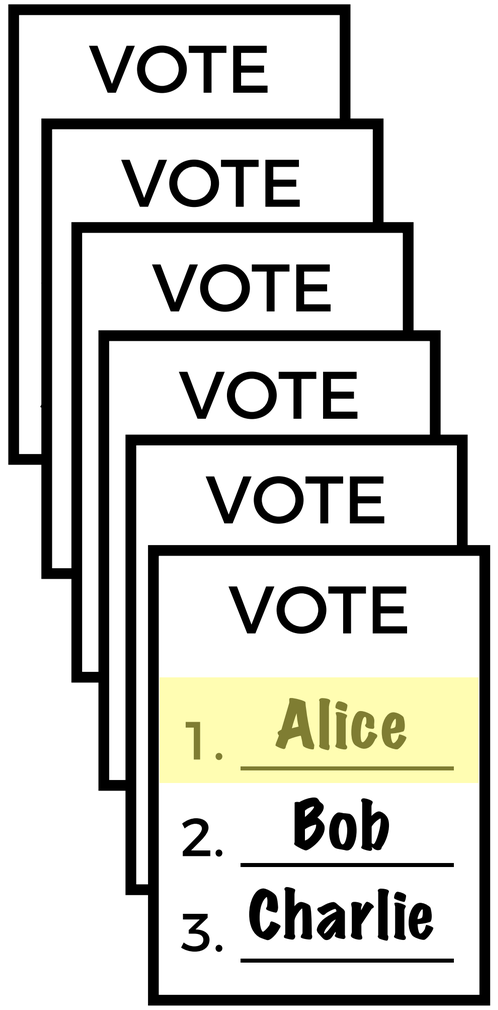


Alice

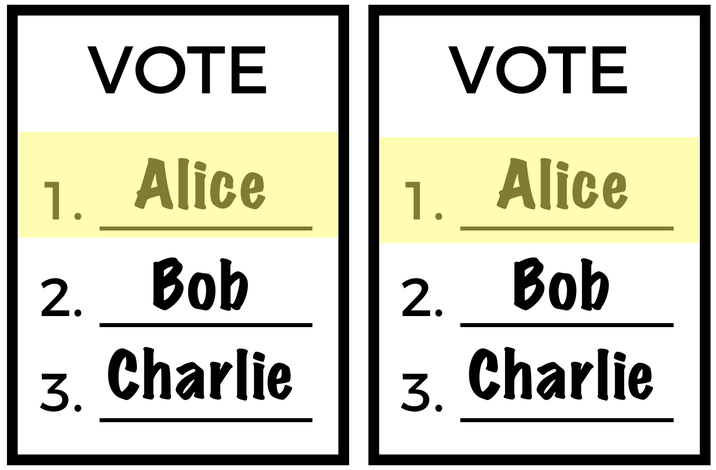


Bob

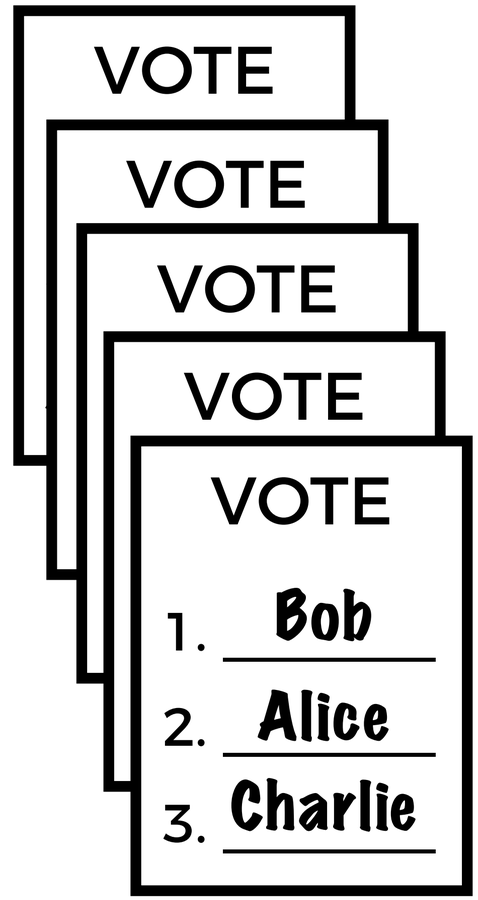
Charlie



Alice

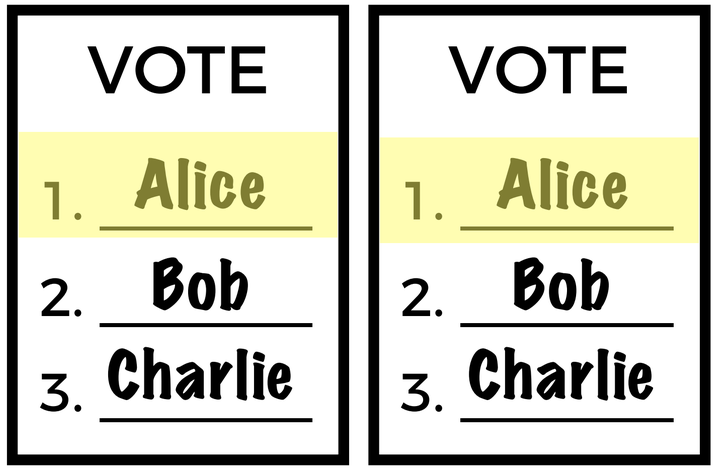


Bob

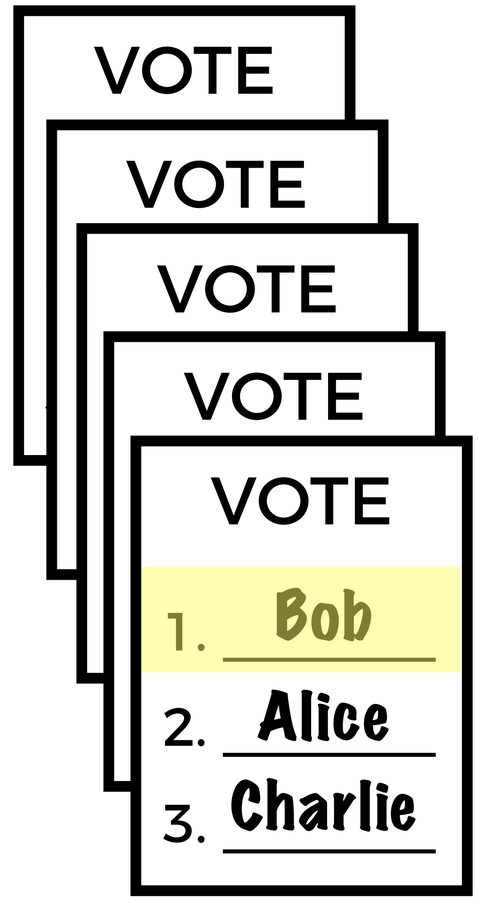


Charlie

Alice



Bob



Charlie

Alice

VOTE

1. Alice
2. Bob
3. Charlie

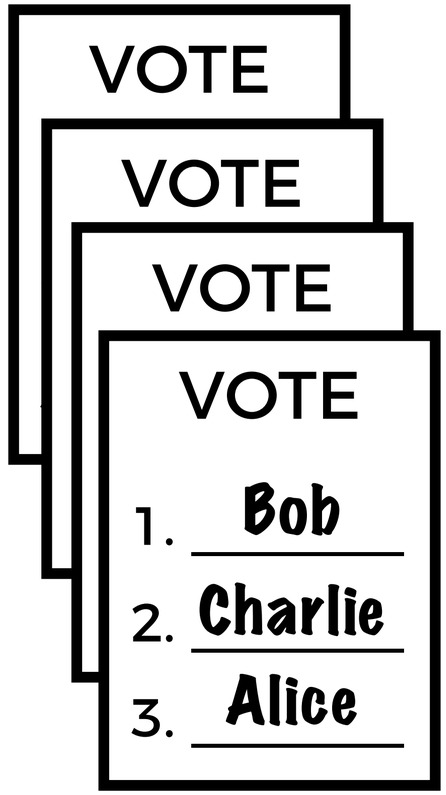
VOTE

1. Alice
2. Bob
3. Charlie

Bob

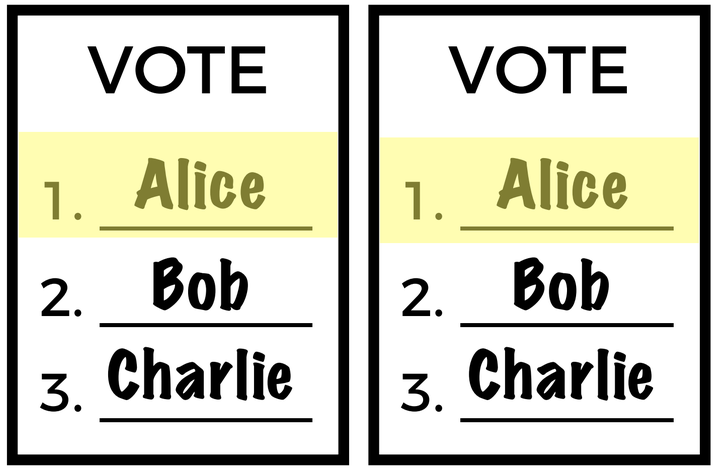
VOTE

1. Bob
2. Alice
3. Charlie

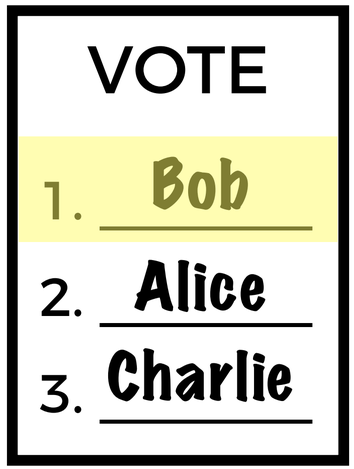


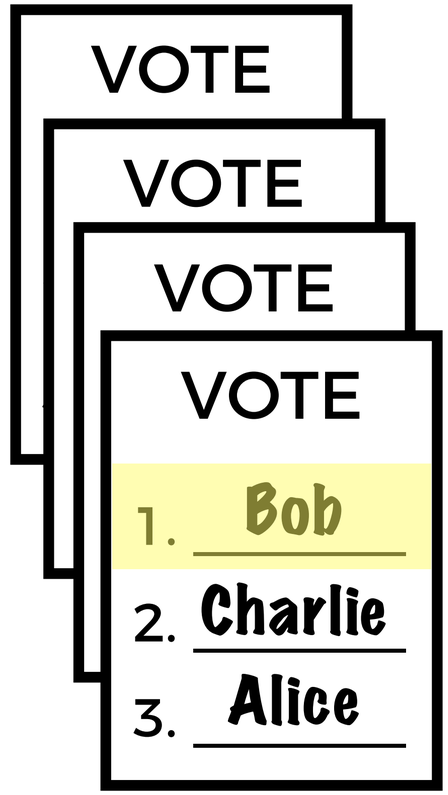
Charlie

Alice



Bob





Charlie

Alice

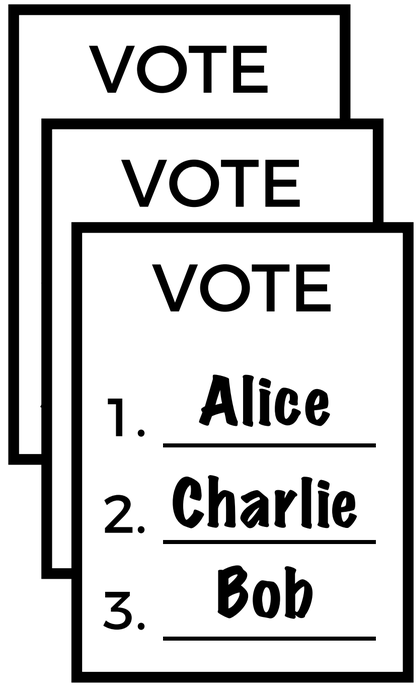
VOTE
1. Alice
2. Bob
3. Charlie

VOTE
1. Alice
2. Bob
3. Charlie

Bob

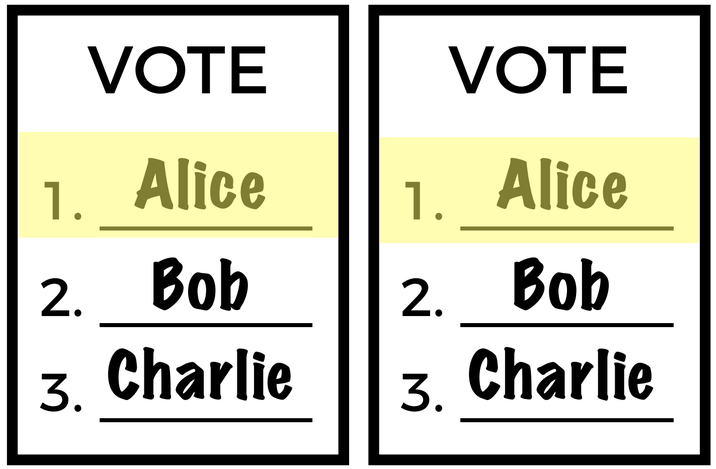
VOTE
1. Bob
2. Alice
3. Charlie

VOTE
1. Bob
2. Charlie
3. Alice

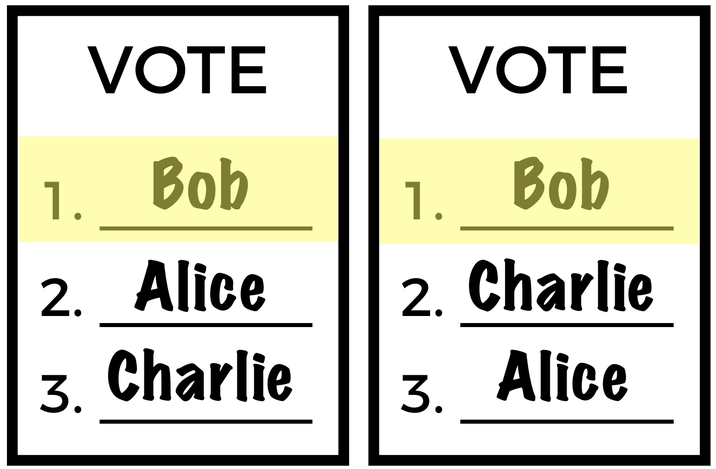


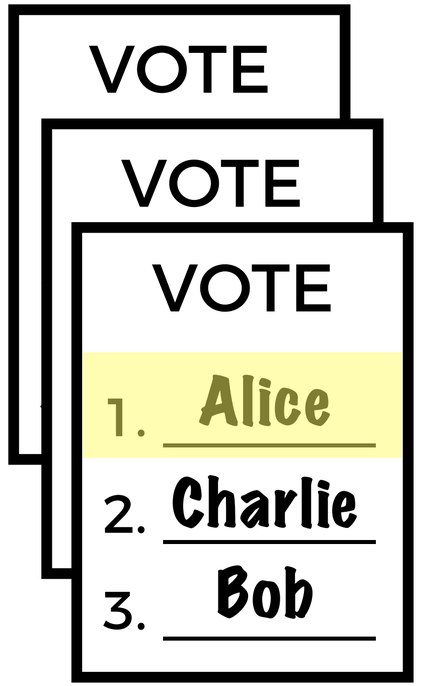
Charlie

Alice



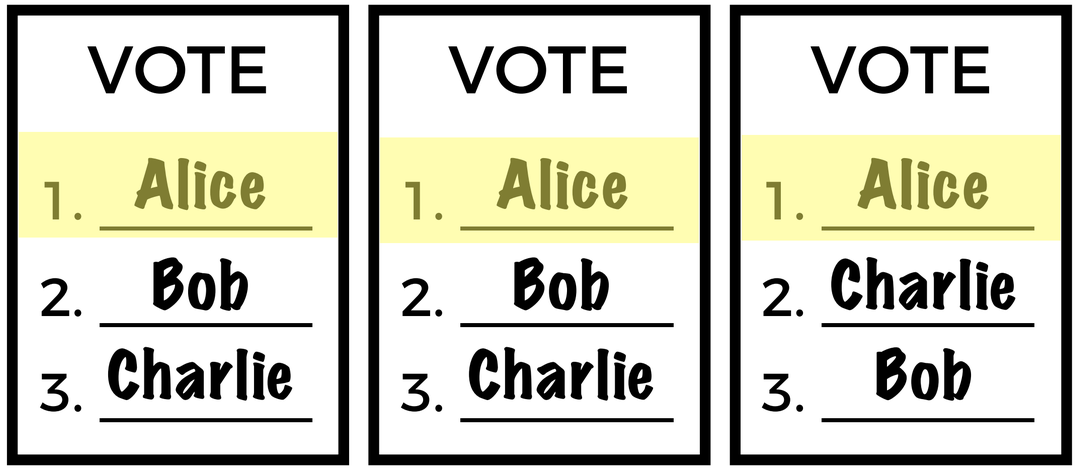
Bob



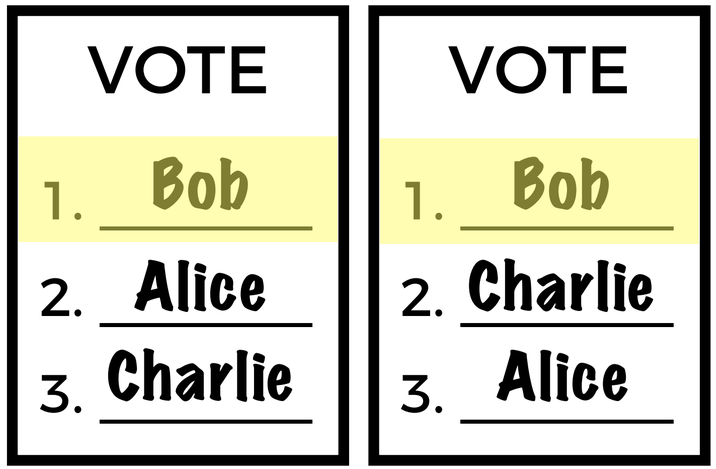


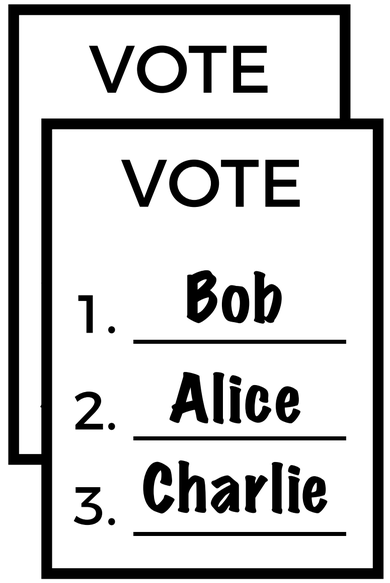
Charlie

Alice



Bob





Charlie

Alice

VOTE
1. Alice
2. Bob
3. Charlie

VOTE
1. Alice
2. Bob
3. Charlie

VOTE
1. Alice
2. Charlie
3. Bob

Bob

VOTE
1. Bob
2. Alice
3. Charlie

VOTE
1. Bob
2. Charlie
3. Alice

VOTE
1. Bob
2. Alice
3. Charlie

Charlie

Alice

VOTE	VOTE	VOTE
1. Alice	1. Alice	1. Alice
2. Bob	2. Bob	2. Charlie
3. Charlie	3. Charlie	3. Bob

Bob

VOTE	VOTE	VOTE
1. Bob	1. Bob	1. Bob
2. Alice	2. Charlie	2. Alice
3. Charlie	3. Alice	3. Charlie

VOTE
1. Charlie
2. Alice
3. Bob

Charlie

Alice

VOTE
1. Alice
2. Bob
3. Charlie

VOTE
1. Alice
2. Bob
3. Charlie

VOTE
1. Alice
2. Charlie
3. Bob

Bob

VOTE
1. Bob
2. Alice
3. Charlie

VOTE
1. Bob
2. Charlie
3. Alice

VOTE
1. Bob
2. Alice
3. Charlie

VOTE
1. Charlie
2. Alice
3. Bob

Charlie

Alice

VOTE
1. Alice
2. Bob
3. Charlie

VOTE
1. Alice
2. Bob
3. Charlie

VOTE
1. Alice
2. Charlie
3. Bob

Bob

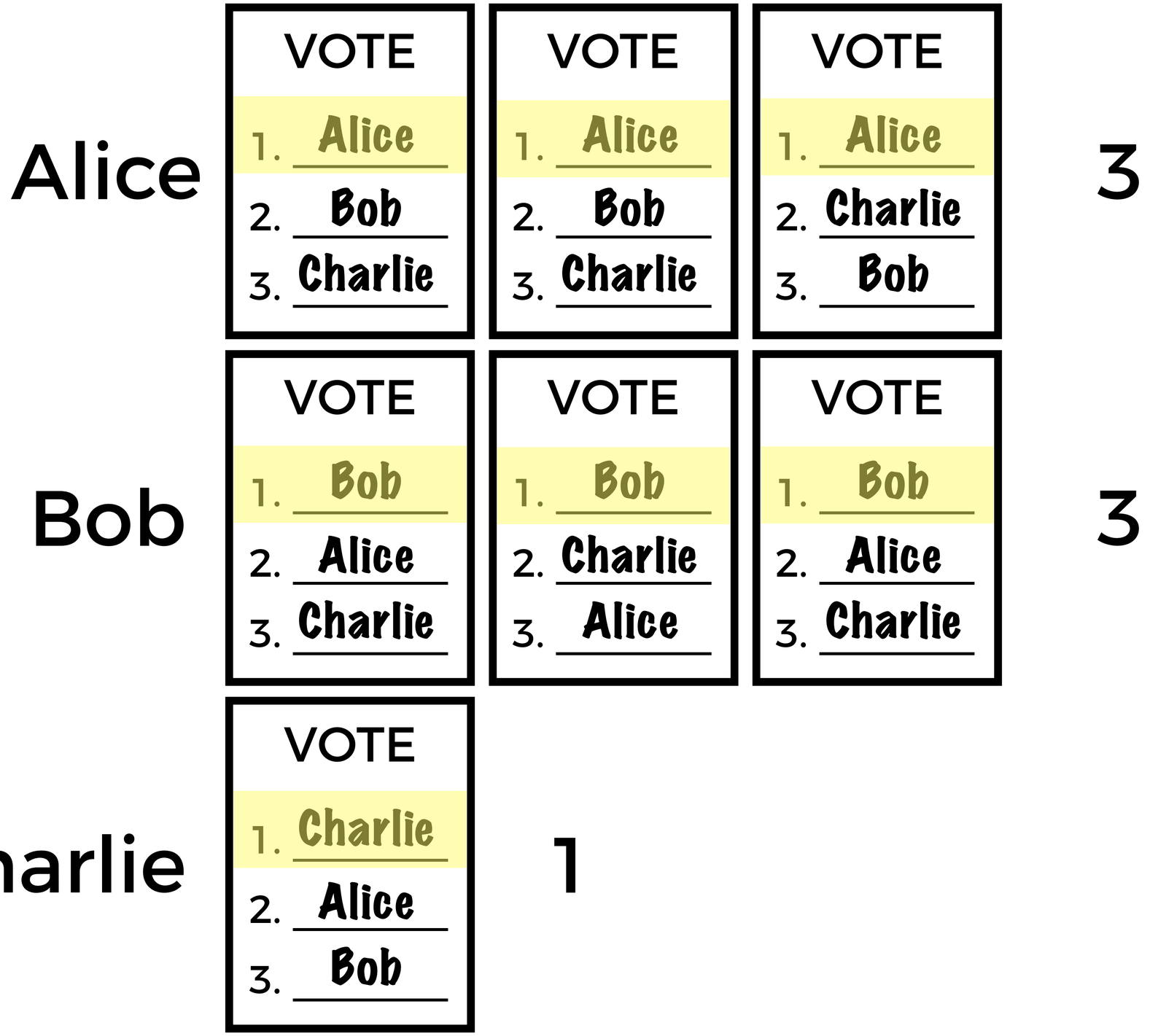
VOTE
1. Bob
2. Alice
3. Charlie

VOTE
1. Bob
2. Charlie
3. Alice

VOTE
1. Bob
2. Alice
3. Charlie

Charlie

VOTE
1. Charlie
2. Alice
3. Bob


Alice
VOTE
1. Alice
2. Bob
3. Charlie
VOTE
1. Alice
2. Bob
3. Charlie
VOTE
1. Alice
2. Charlie
3. Bob
3
Bob
VOTE
1. Bob
2. Alice
3. Charlie
VOTE
1. Bob
2. Charlie
3. Alice
VOTE
1. Bob
2. Alice
3. Charlie
3
Charlie
VOTE
1. Charlie
2. Alice
3. Bob
1

Alice

VOTE	VOTE	VOTE
1. Alice	1. Alice	1. Alice
2. Bob	2. Bob	2. Charlie
3. Charlie	3. Charlie	3. Bob

3

Bob

VOTE	VOTE	VOTE
1. Bob	1. Bob	1. Bob
2. Alice	2. Charlie	2. Alice
3. Charlie	3. Alice	3. Charlie

3

VOTE
1. Charlie
2. Alice
3. Bob

Alice

VOTE	VOTE	VOTE
1. Alice	1. Alice	1. Alice
2. Bob	2. Bob	2. Charlie
3. Charlie	3. Charlie	3. Bob

3

Bob

VOTE	VOTE	VOTE
1. Bob	1. Bob	1. Bob
2. Alice	2. Charlie	2. Alice
3. Charlie	3. Alice	3. Charlie

3

VOTE
1. Charlie
2. Alice
3. Bob

Alice

VOTE
1. Alice
2. Bob
3. Charlie

VOTE
1. Alice
2. Bob
3. Charlie

VOTE
1. Alice
2. Charlie
3. Bob

VOTE
1. Charlie
2. Alice
3. Bob

4

Bob

VOTE
1. Bob
2. Alice
3. Charlie

VOTE
1. Bob
2. Charlie
3. Alice

VOTE
1. Bob
2. Alice
3. Charlie

3

Alice

VOTE
1. Alice
2. Bob
3. Charlie

VOTE
1. Alice
2. Bob
3. Charlie

VOTE
1. Alice
2. Charlie
3. Bob

VOTE
1. Charlie
2. Alice
3. Bob

4

Bob

VOTE
1. Bob
2. Alice
3. Charlie

VOTE
1. Bob
2. Charlie
3. Alice

VOTE
1. Bob
2. Alice
3. Charlie

3

Representing Candidates

name	Alice	Bob	Charlie	Dave
votes	3	2	4	0
eliminated	false	false	false	true

Representing Candidates

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

Representing Candidates

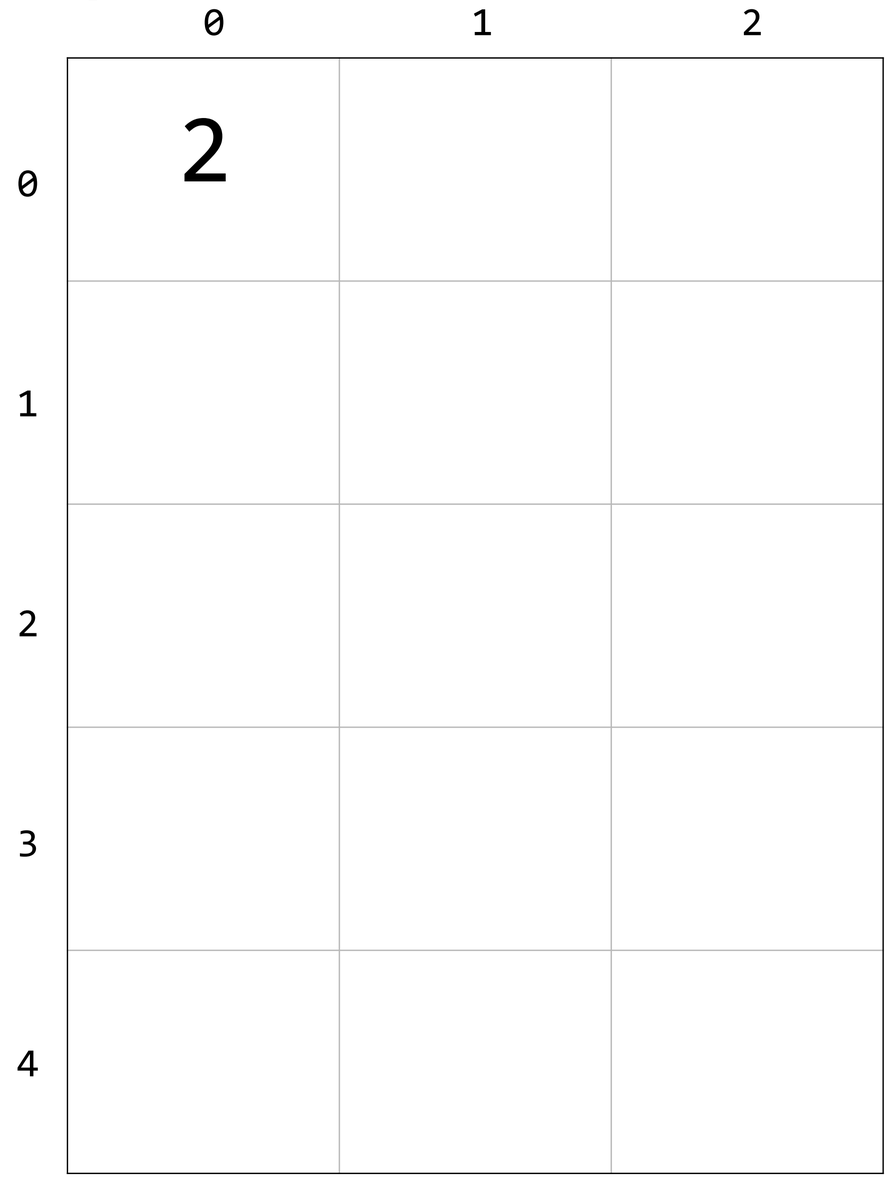
```
candidate candidates[MAX_CANDIDATES];
```

Representing Preferences

```
int preferences[MAX_VOTERS][MAX_CANDIDATES];
```

`preferences[i][j]` is the candidate index of voter `i`'s preference `j`.

preferences



candidates

name	Alice	Bob	Charlie
votes	0	0	0
eliminated	false	false	false

```
preferences[0][0] = 2;
```

First voter's top preference is Charlie.

preferences

	0	1	2
0	2	0	
1			
2			
3			
4			

candidates

name	Alice	Bob	Charlie
votes	0	0	0
eliminated	false	false	false

```
preferences[0][1] = 0;
```

First voter's second preference is Alice.

This is CS50.