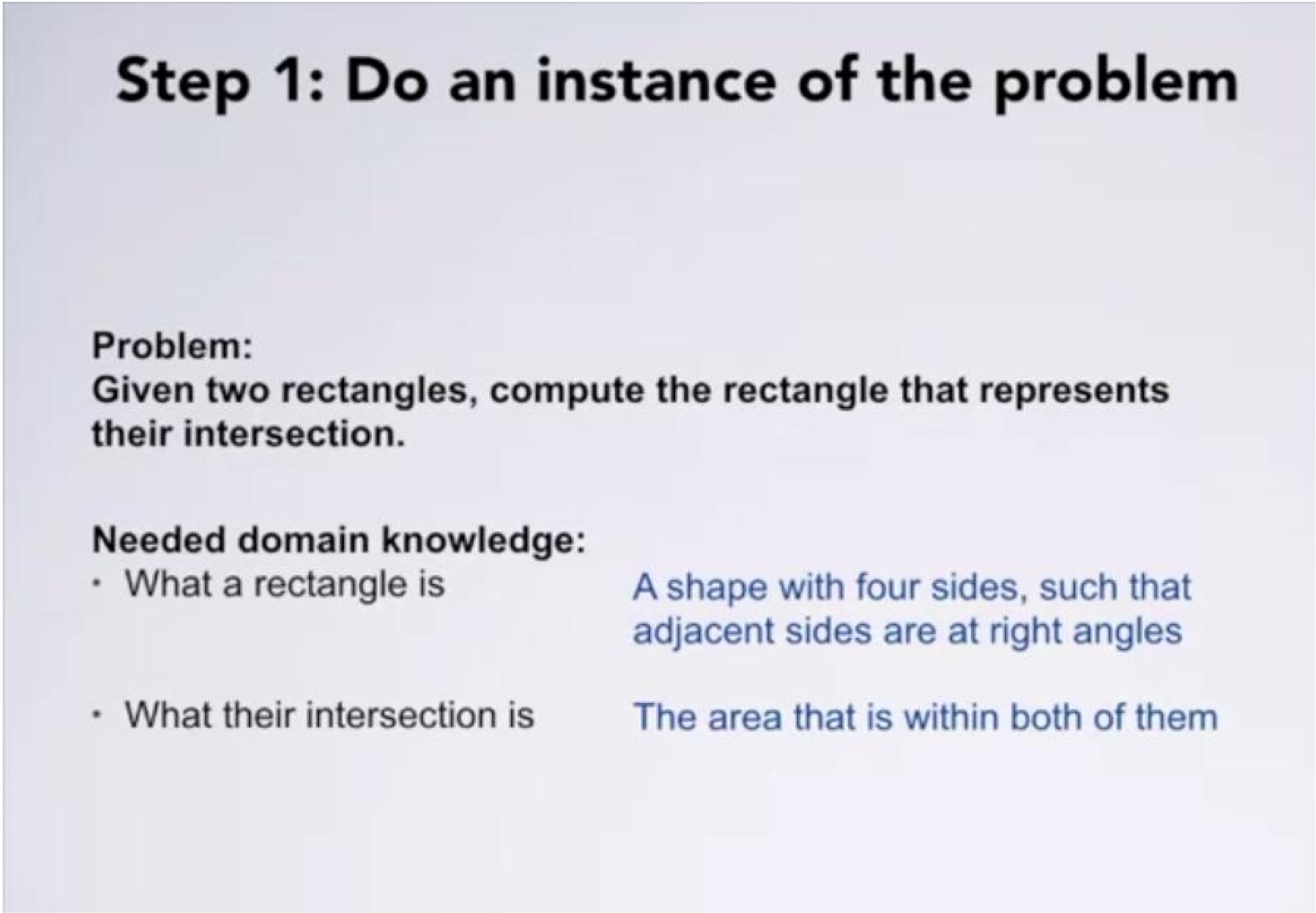


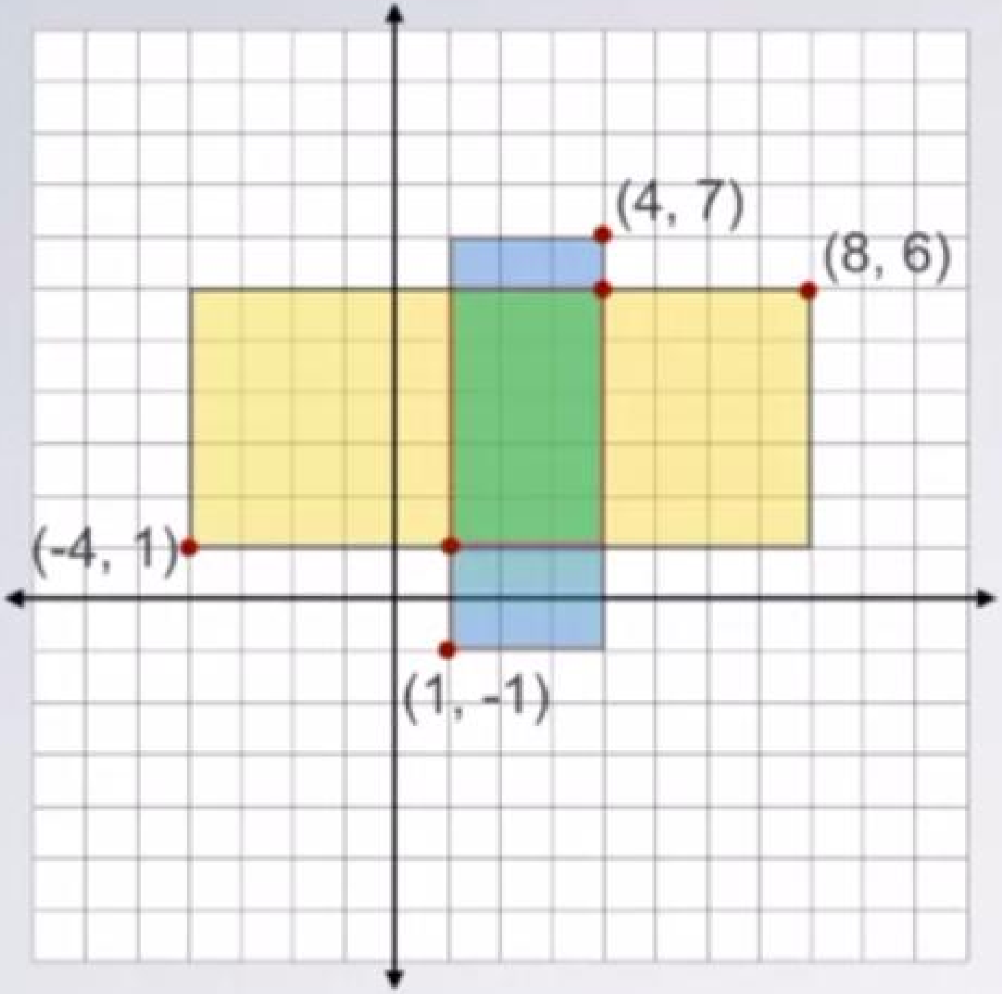
Step 1: Do an instance of the problem

Problem:
Given two rectangles, compute the rectangle that represents their intersection.

Needed domain knowledge:

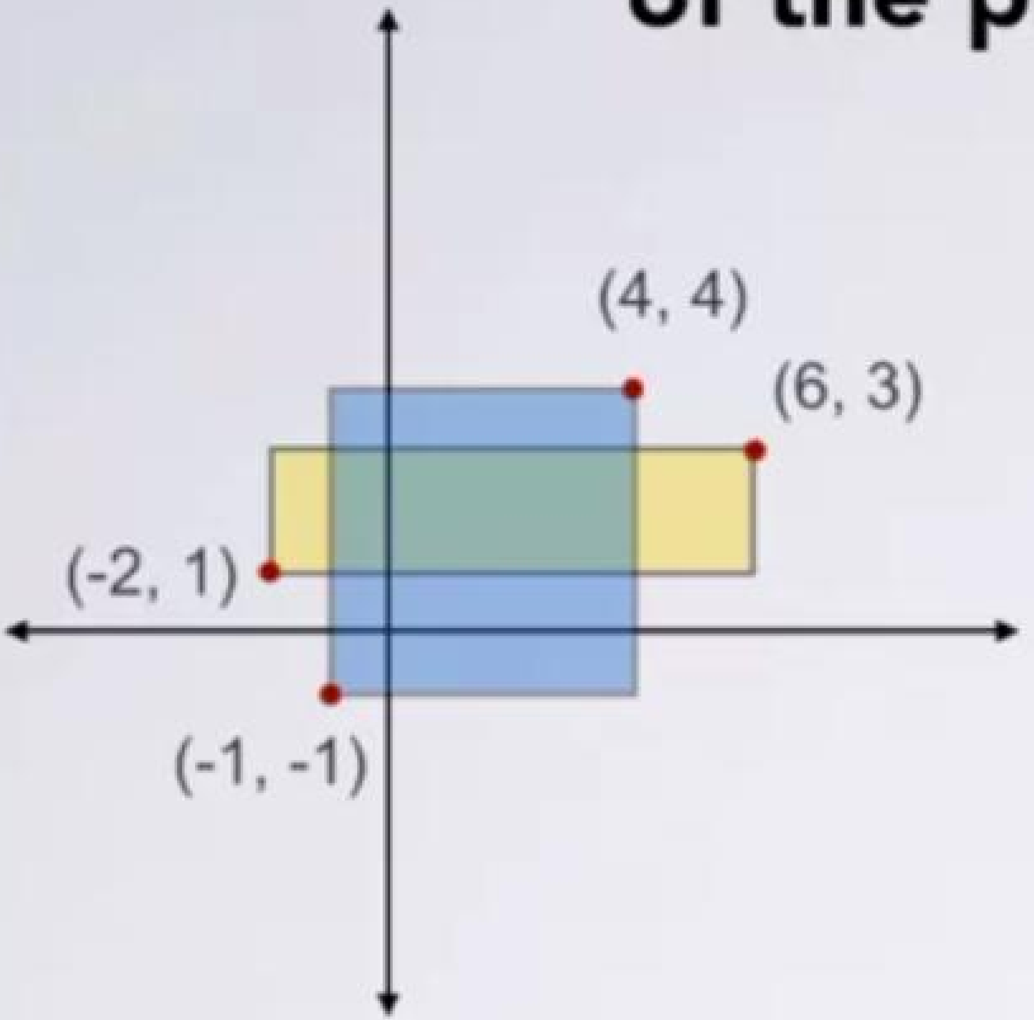
- What a rectangle is — A shape with four sides, such that adjacent sides are at right angles
- What their intersection is — The area that is within both of them

Step 1: Do an instance of the problem

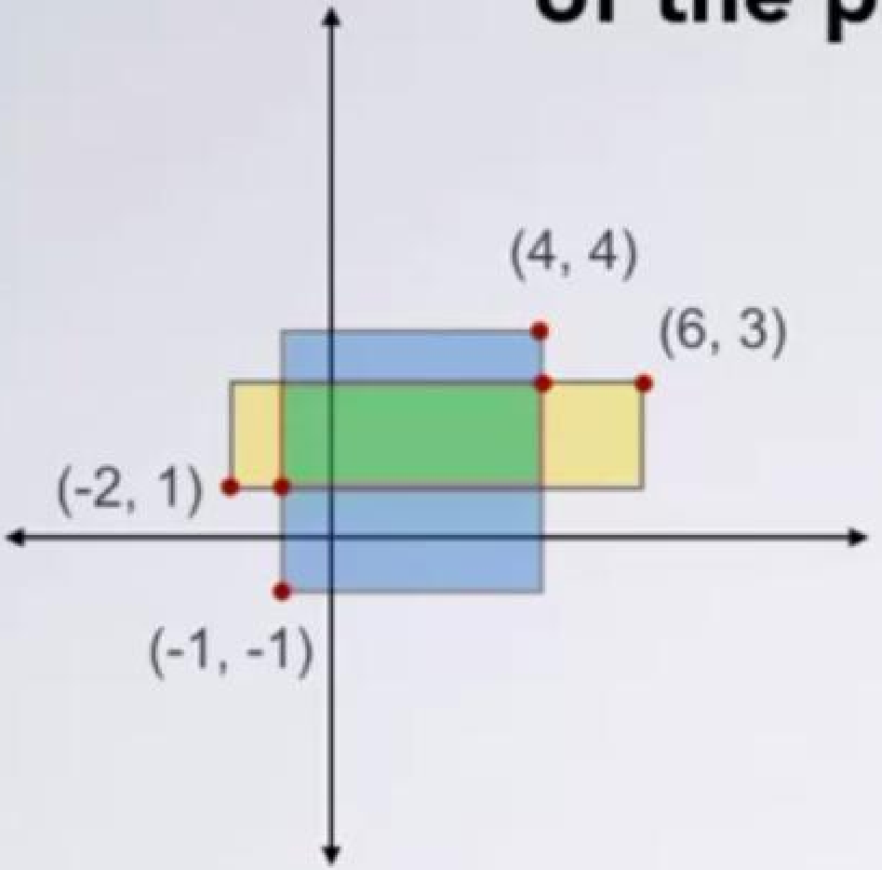


Intersection is the rectangle from (1,1) to (4, 6)

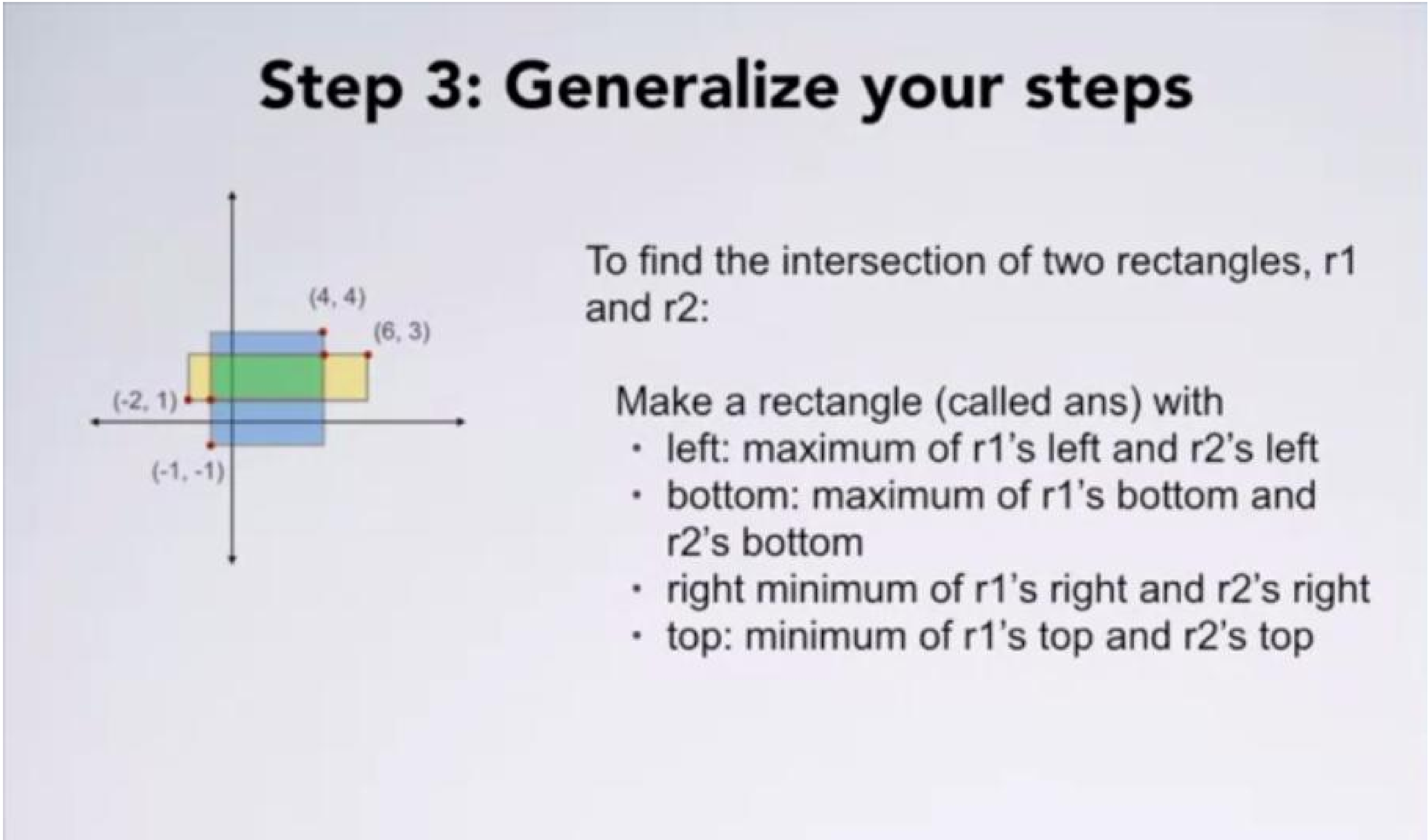
Step 1: Do an(other) instance of the problem



Step 1: Do an(other) instance of the problem



Intersection is the rectangle from (-1,1) to (4, 3)

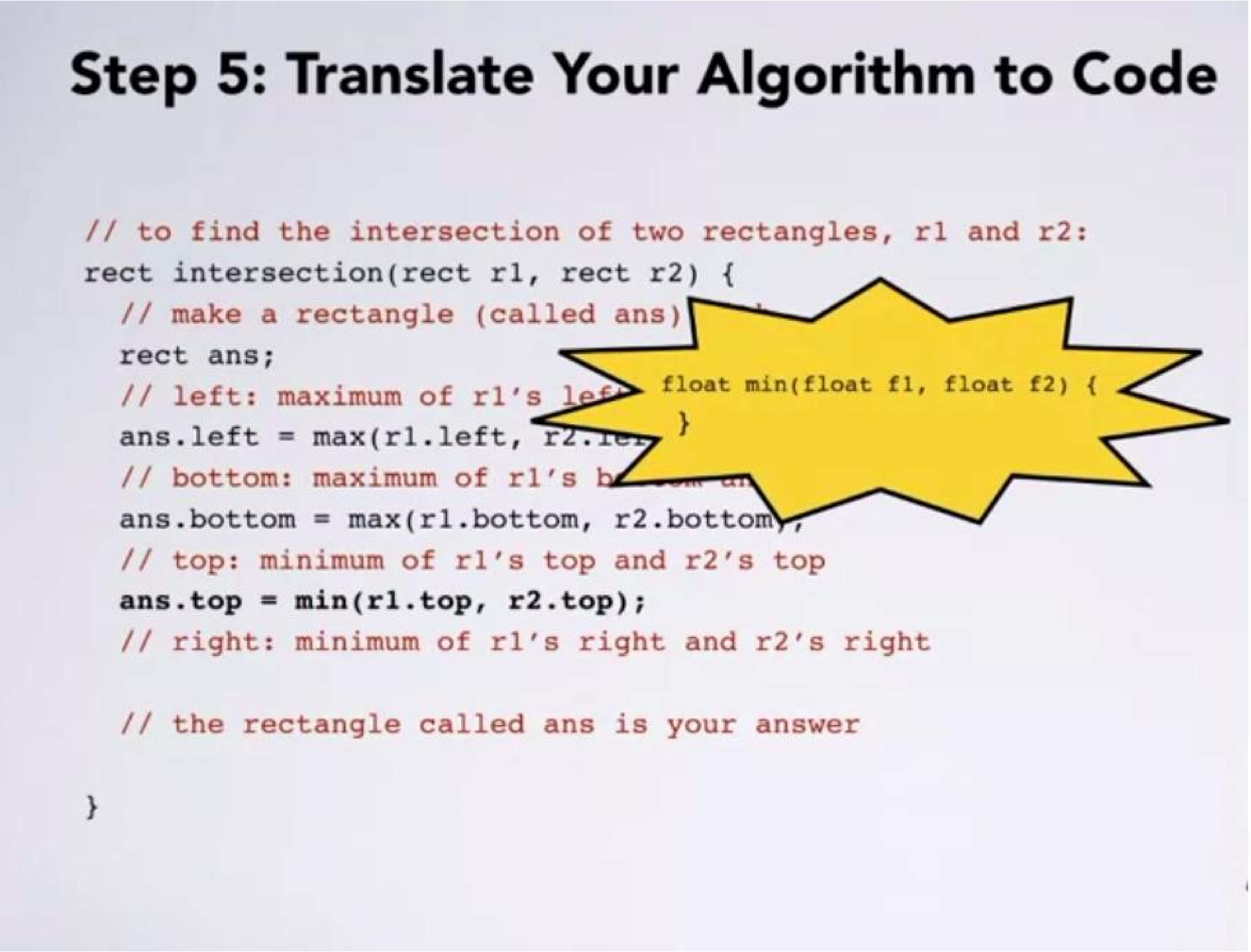


Step 3: Generalize your steps

To find the intersection of two rectangles, r1 and r2:

Make a rectangle (called ans) with

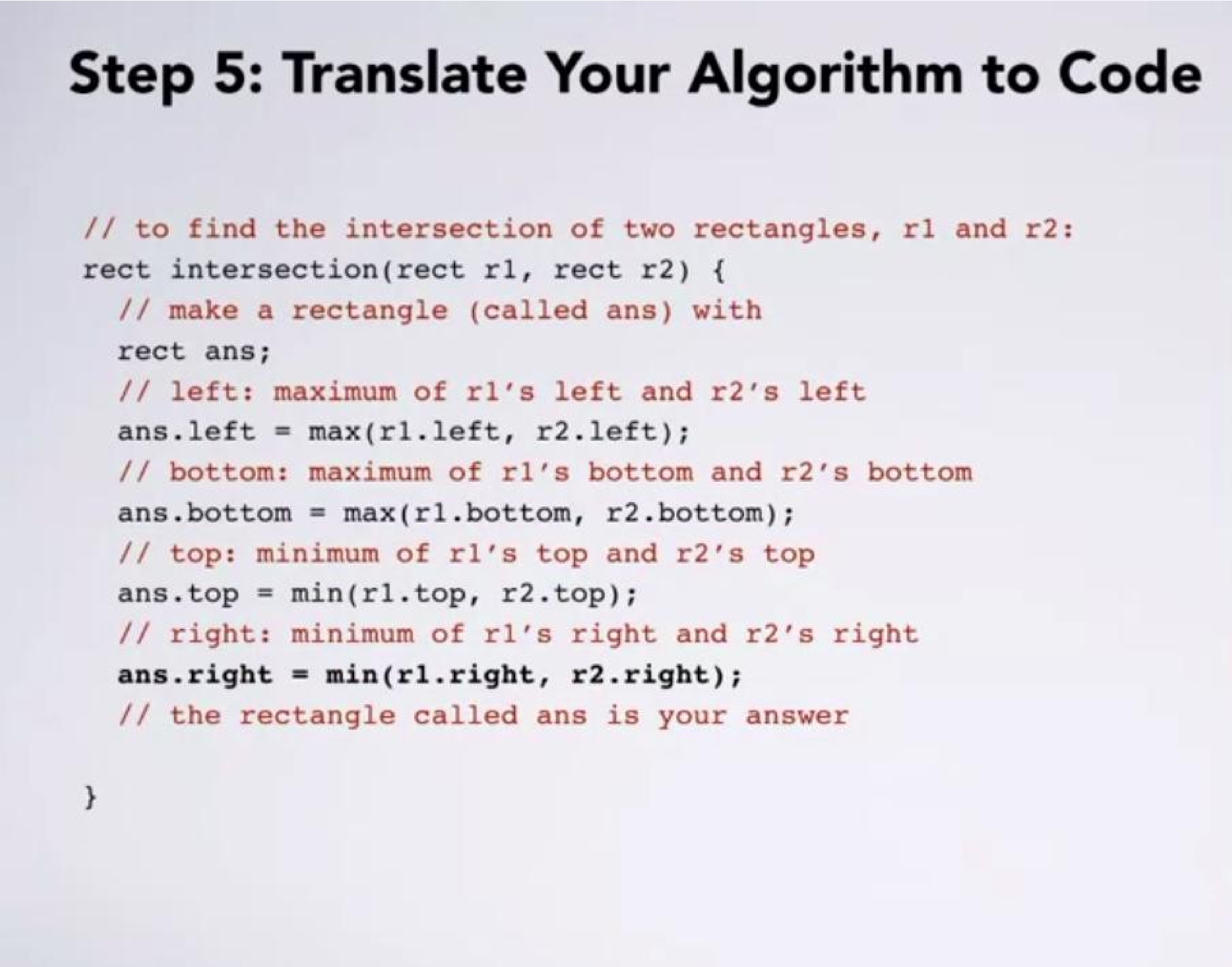
- left: maximum of r1's left and r2's left
- bottom: maximum of r1's bottom and r2's bottom
- right minimum of r1's right and r2's right
- top: minimum of r1's top and r2's top



Step 5: Translate Your Algorithm to Code

```
// to find the intersection of two rectangles, r1 and r2:
rect intersection(rect r1, rect r2) {
  // make a rectangle (called ans)
  rect ans;
  // left: maximum of r1's lef
  ans.left = max(r1.left, r2.le
  // bottom: maximum of r1's b
  ans.bottom = max(r1.bottom, r2.bottom);
  // top: minimum of r1's top and r2's top
  ans.top = min(r1.top, r2.top);
  // right: minimum of r1's right and r2's right

  // the rectangle called ans is your answer

}
```

```
float min(float f1, float f2) {
 }
```



Step 5: Translate Your Algorithm to Code

```
// to find the intersection of two rectangles, r1 and r2:
rect intersection(rect r1, rect r2) {
  // make a rectangle (called ans) with
  rect ans;
  // left: maximum of r1's left and r2's left
  ans.left = max(r1.left, r2.left);
  // bottom: maximum of r1's bottom and r2's bottom
  ans.bottom = max(r1.bottom, r2.bottom);
  // top: minimum of r1's top and r2's top
  ans.top = min(r1.top, r2.top);
  // right: minimum of r1's right and r2's right
  ans.right = min(r1.right, r2.right);
  // the rectangle called ans is your answer

}
```

Step 1: Do an Instance of the problem

Given an integer N, determine if N is prime.

Is 7 prime? "Yes, I just know this" (not helpful)

May be hard to see past "I just know this," if so:

- Think about how you would convince someone this is right
- Think about a harder problem to see step-by-step approach

Is 29393 prime?

Step 2: Write Down Exactly What You Did

Is 29393 prime?

29393/2 = 14696 remainder 1   checked if 29393 mod 2 is 0 (no)

29393/3 = 9797 remainder 2

29393/4 = 7348 remainder 1

29393/5 = 5878 remainder 3

29393/6 = 4898 remainder 5

29393/7 = 4199 remainder 0

- answer "no"

Step 2: Write Down Exactly What You Did

Is 29393 prime?

29393/2 = 14696 remainder 1	checked if 29393 mod 2 is 0 (no)
29393/3 = 9797 remainder 2	checked if 29393 mod 3 is 0 (no)
29393/4 = 7348 remainder 1	checked if 29393 mod 4 is 0 (no)
29393/5 = 5878 remainder 3	checked if 29393 mod 5 is 0 (no)
29393/6 = 4898 remainder 5	checked if 29393 mod 6 is 0 (no)
29393/7 = 4199 remainder 0	checked if 29393 mod 7 is 0 (yes)

➤ answer "no"

Step 1: Do an Instance of the problem

Is 7 prime?

7/2 = 3 remainder 1

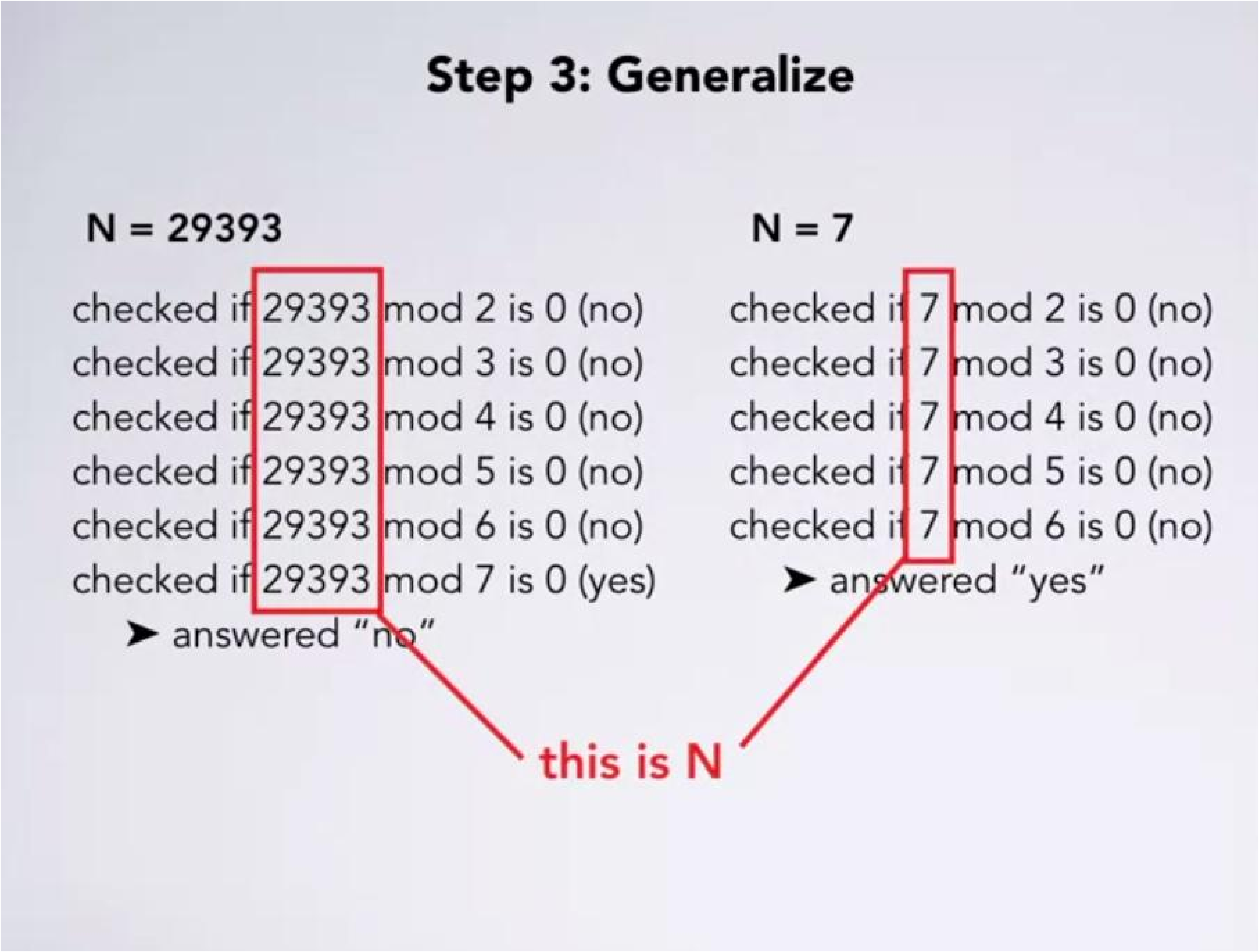
7/3 = 2 remainder 1

7/4 = 1 remainder 3

7/5 = 1 remainder 2

7/6 = 1 remainder 1

- ➤ answer "yes"



Step 3: Generalize

N = 29393

checked if 29393 mod 2 is 0 (no)

checked if 29393 mod 3 is 0 (no)

checked if 29393 mod 4 is 0 (no)

checked if 29393 mod 5 is 0 (no)

checked if 29393 mod 6 is 0 (no)

checked if 29393 mod 7 is 0 (yes)

➤ answered "no"

N = 7

checked if 7 mod 2 is 0 (no)

checked if 7 mod 3 is 0 (no)

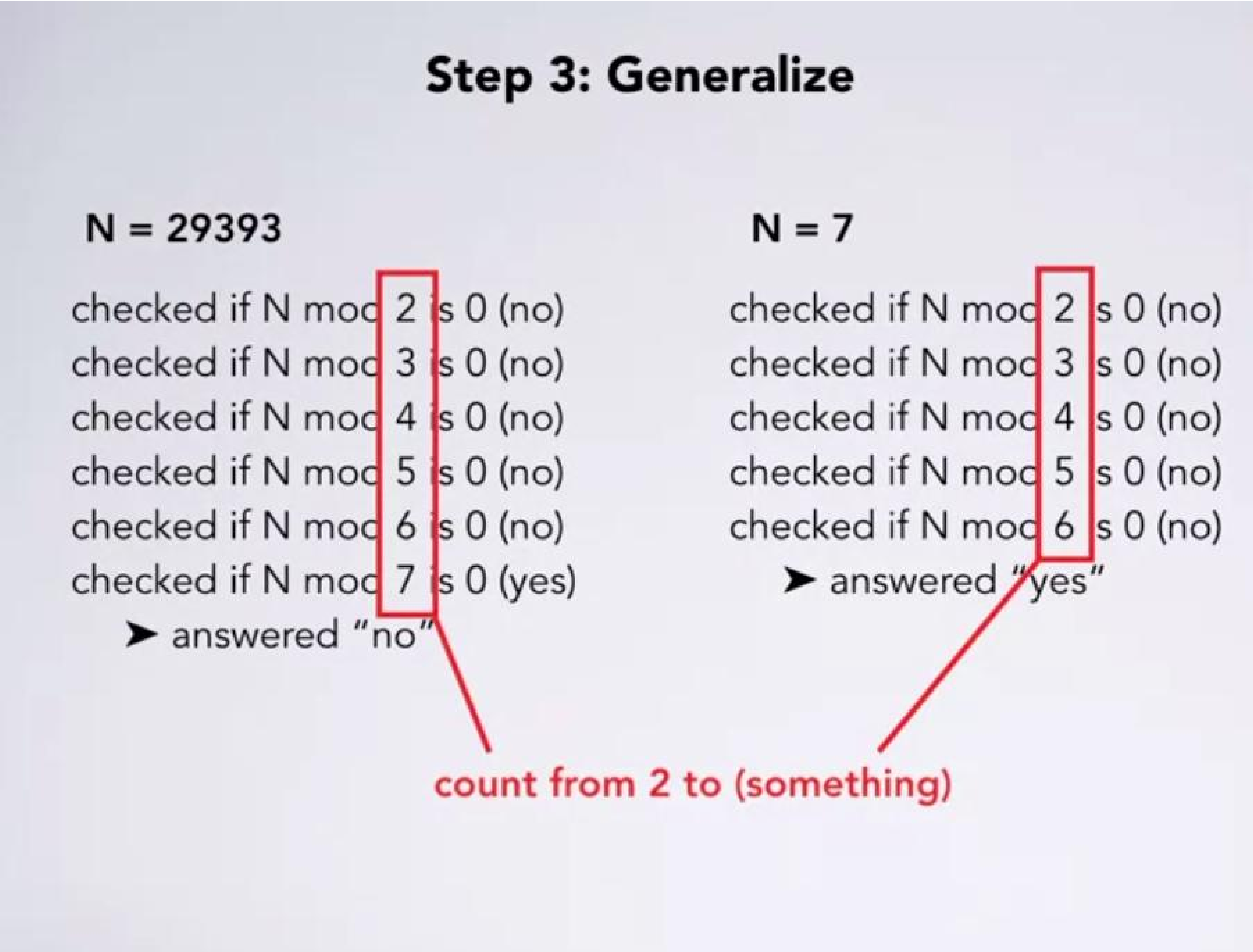
checked if 7 mod 4 is 0 (no)

checked if 7 mod 5 is 0 (no)

checked if 7 mod 6 is 0 (no)

➤ answered "yes"

this is N



Step 3: Generalize

N = 29393

checked if N mod 2 is 0 (no)
checked if N mod 3 is 0 (no)
checked if N mod 4 is 0 (no)
checked if N mod 5 is 0 (no)
checked if N mod 6 is 0 (no)
checked if N mod 7 is 0 (yes)

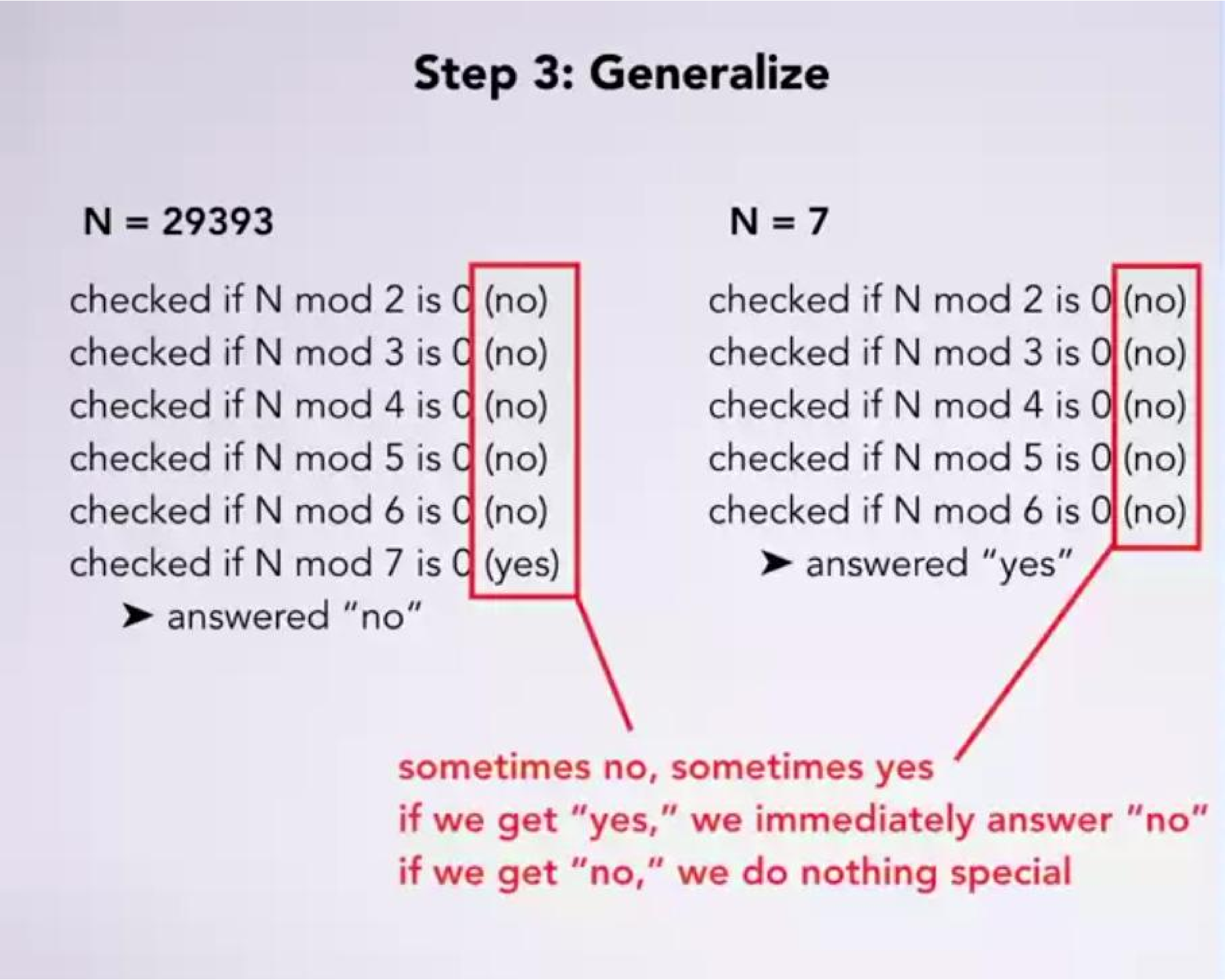
➤ answered "no"

N = 7

checked if N mod 2 is 0 (no)
checked if N mod 3 is 0 (no)
checked if N mod 4 is 0 (no)
checked if N mod 5 is 0 (no)
checked if N mod 6 is 0 (no)

➤ answered "yes"

count from 2 to (something)



Step 3: Generalize

N = 29393

checked if N mod 2 is 0 (no)
checked if N mod 3 is 0 (no)
checked if N mod 4 is 0 (no)
checked if N mod 5 is 0 (no)
checked if N mod 6 is 0 (no)
checked if N mod 7 is 0 (yes)

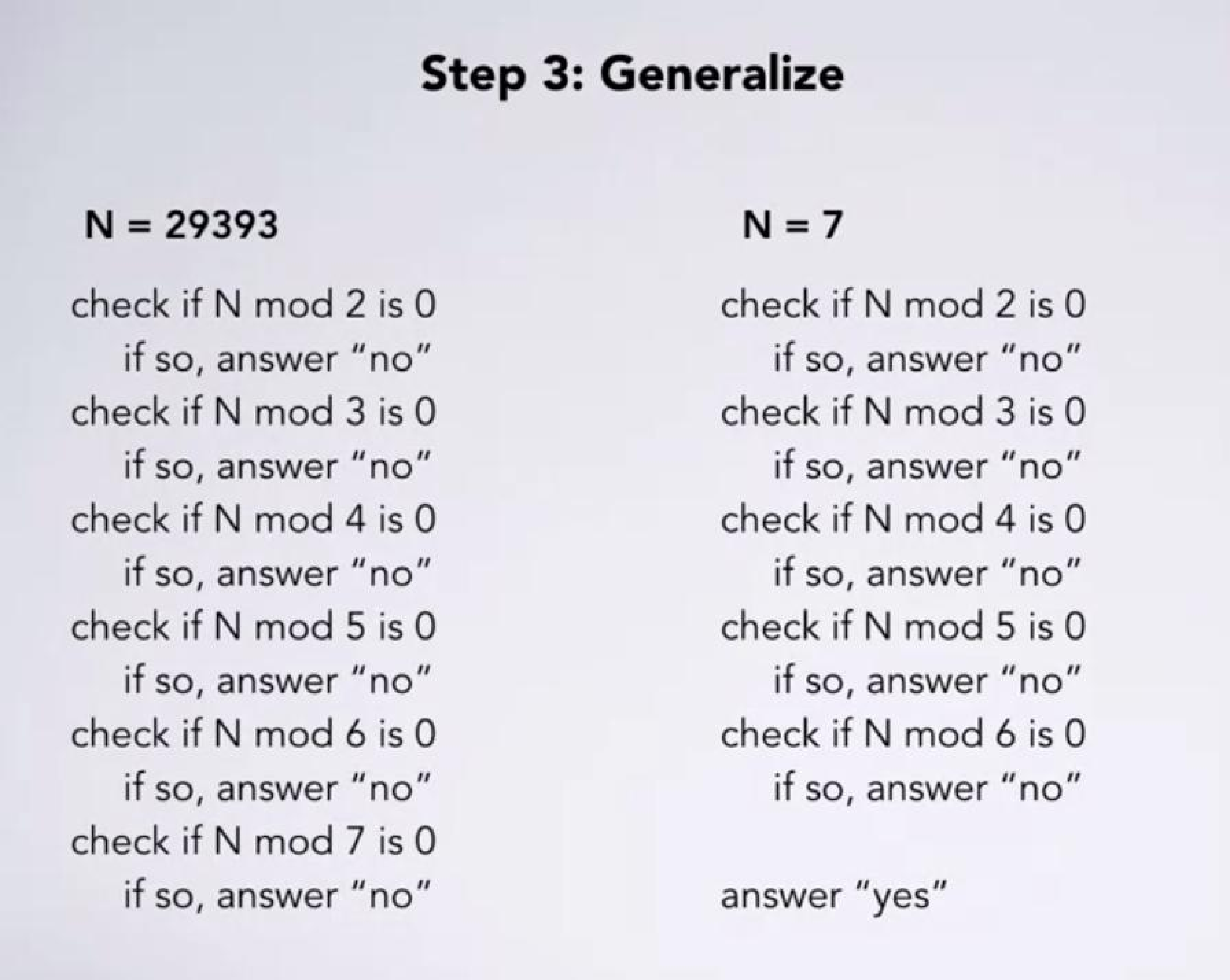
➤ answered "no"

N = 7

checked if N mod 2 is 0 (no)
checked if N mod 3 is 0 (no)
checked if N mod 4 is 0 (no)
checked if N mod 5 is 0 (no)
checked if N mod 6 is 0 (no)

➤ answered "yes"

sometimes no, sometimes yes
if we get "yes," we immediately answer "no"
if we get "no," we do nothing special



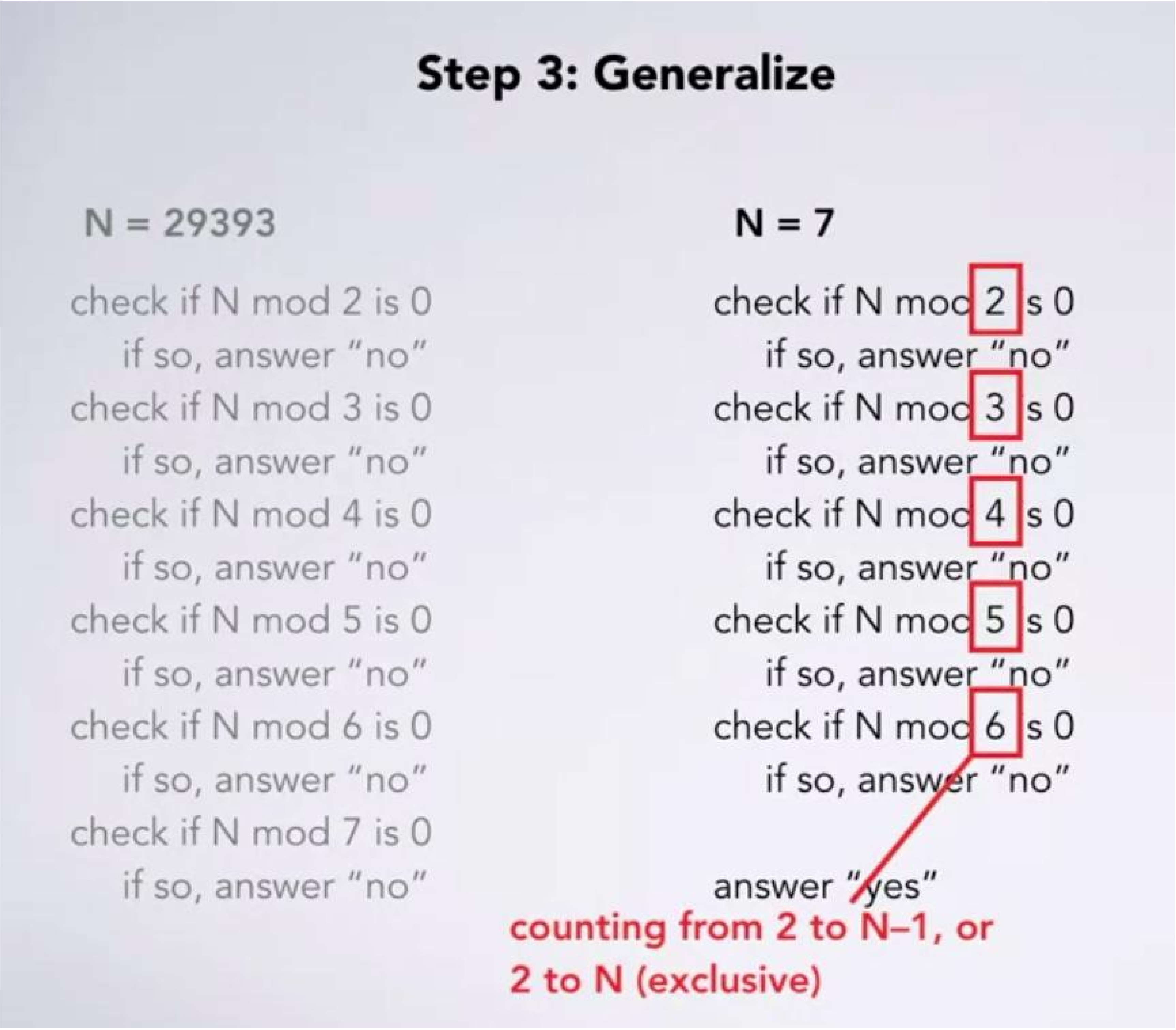
Step 3: Generalize

N = 29393

```
check if N mod 2 is 0
    if so, answer “no”
check if N mod 3 is 0
    if so, answer “no”
check if N mod 4 is 0
    if so, answer “no”
check if N mod 5 is 0
    if so, answer “no”
check if N mod 6 is 0
    if so, answer “no”
check if N mod 7 is 0
    if so, answer “no”
```

N = 7

```
check if N mod 2 is 0
    if so, answer “no”
check if N mod 3 is 0
    if so, answer “no”
check if N mod 4 is 0
    if so, answer “no”
check if N mod 5 is 0
    if so, answer “no”
check if N mod 6 is 0
    if so, answer “no”

answer “yes”
```



Step 3: Generalize

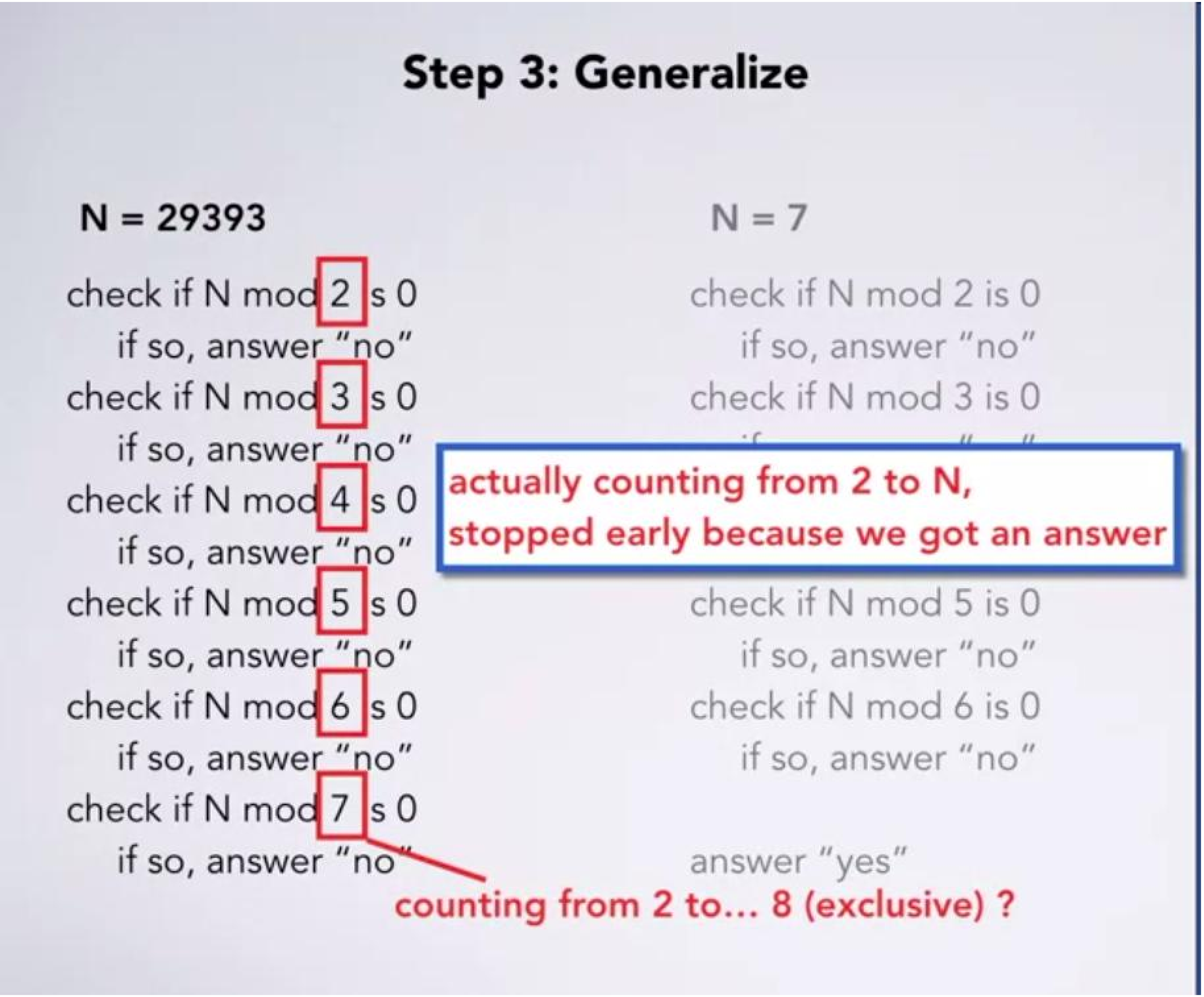
N = 29393

```
check if N mod 2 is 0
    if so, answer "no"
check if N mod 3 is 0
    if so, answer "no"
check if N mod 4 is 0
    if so, answer "no"
check if N mod 5 is 0
    if so, answer "no"
check if N mod 6 is 0
    if so, answer "no"
check if N mod 7 is 0
    if so, answer "no"
```

N = 7

```
check if N mod 2 is 0
    if so, answer "no"
check if N mod 3 is 0
    if so, answer "no"
check if N mod 4 is 0
    if so, answer "no"
check if N mod 5 is 0
    if so, answer "no"
check if N mod 6 is 0
    if so, answer "no"

answer "yes"
```

counting from 2 to N–1, or
2 to N (exclusive)



Step 3: Generalize

N = 29393

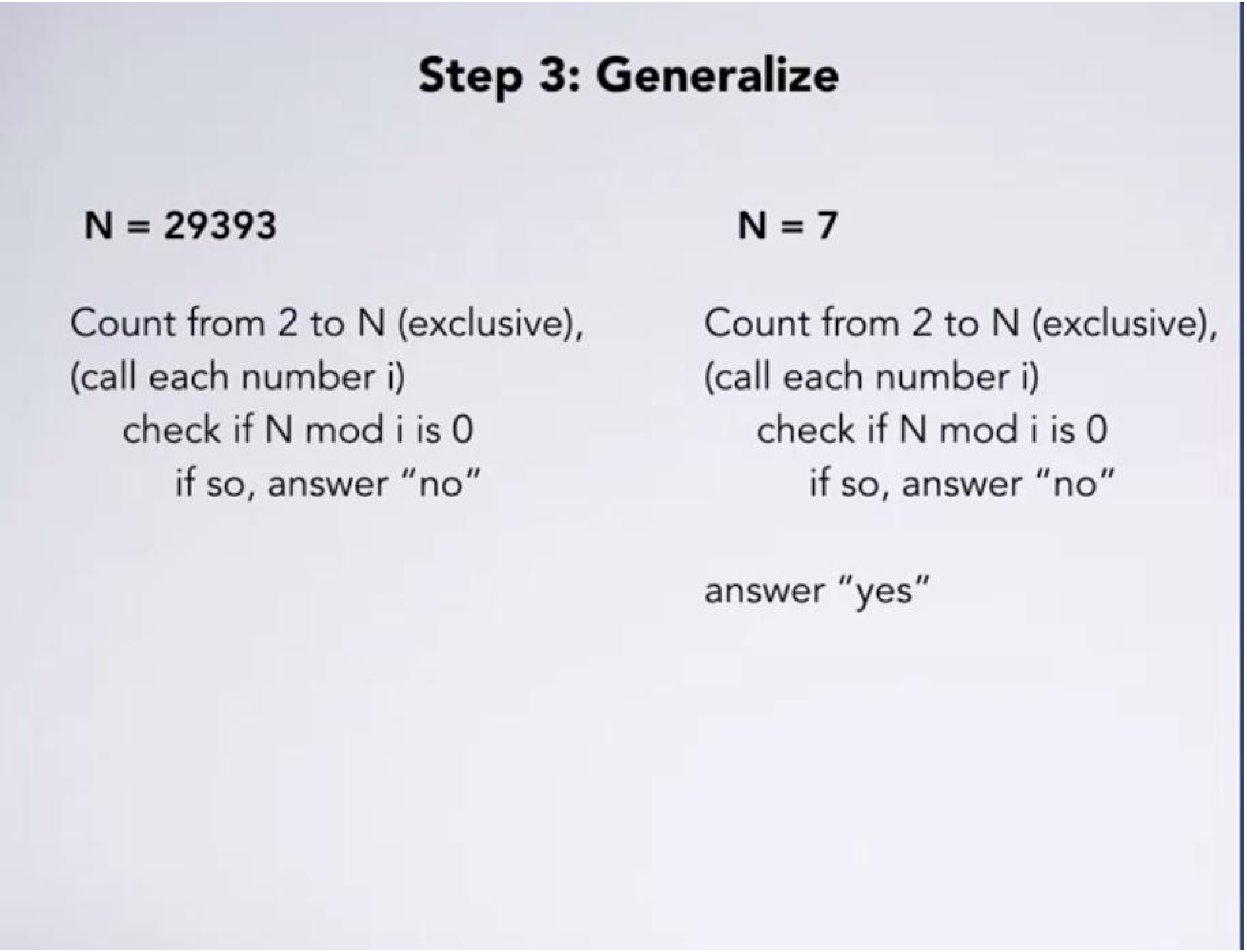
```
check if N mod 2 is 0
    if so, answer "no"
check if N mod 3 is 0
    if so, answer "no"
check if N mod 4 is 0
    if so, answer "no"
check if N mod 5 is 0
    if so, answer "no"
check if N mod 6 is 0
    if so, answer "no"
check if N mod 7 is 0
    if so, answer "no"
```

N = 7

```
check if N mod 2 is 0
    if so, answer "no"
check if N mod 3 is 0
check if N mod 5 is 0
    if so, answer "no"
check if N mod 6 is 0
    if so, answer "no"
answer "yes"
```

actually counting from 2 to N, stopped early because we got an answer

counting from 2 to... 8 (exclusive) ?



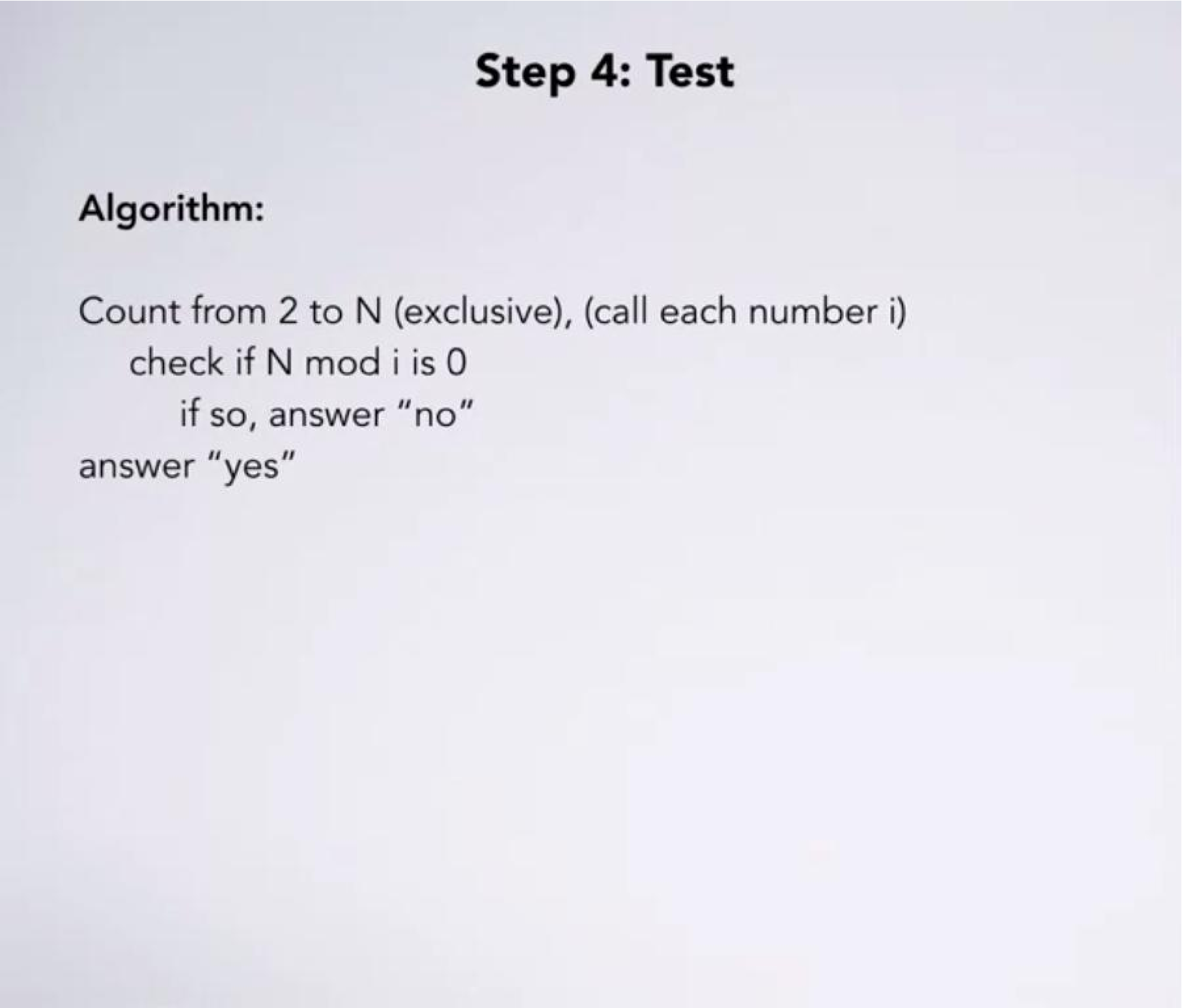
Step 3: Generalize

N = 29393

```
Count from 2 to N (exclusive),
(call each number i)
    check if N mod i is 0
        if so, answer "no"
```

N = 7

```
Count from 2 to N (exclusive),
(call each number i)
    check if N mod i is 0
        if so, answer "no"

answer "yes"
```



Step 4: Test

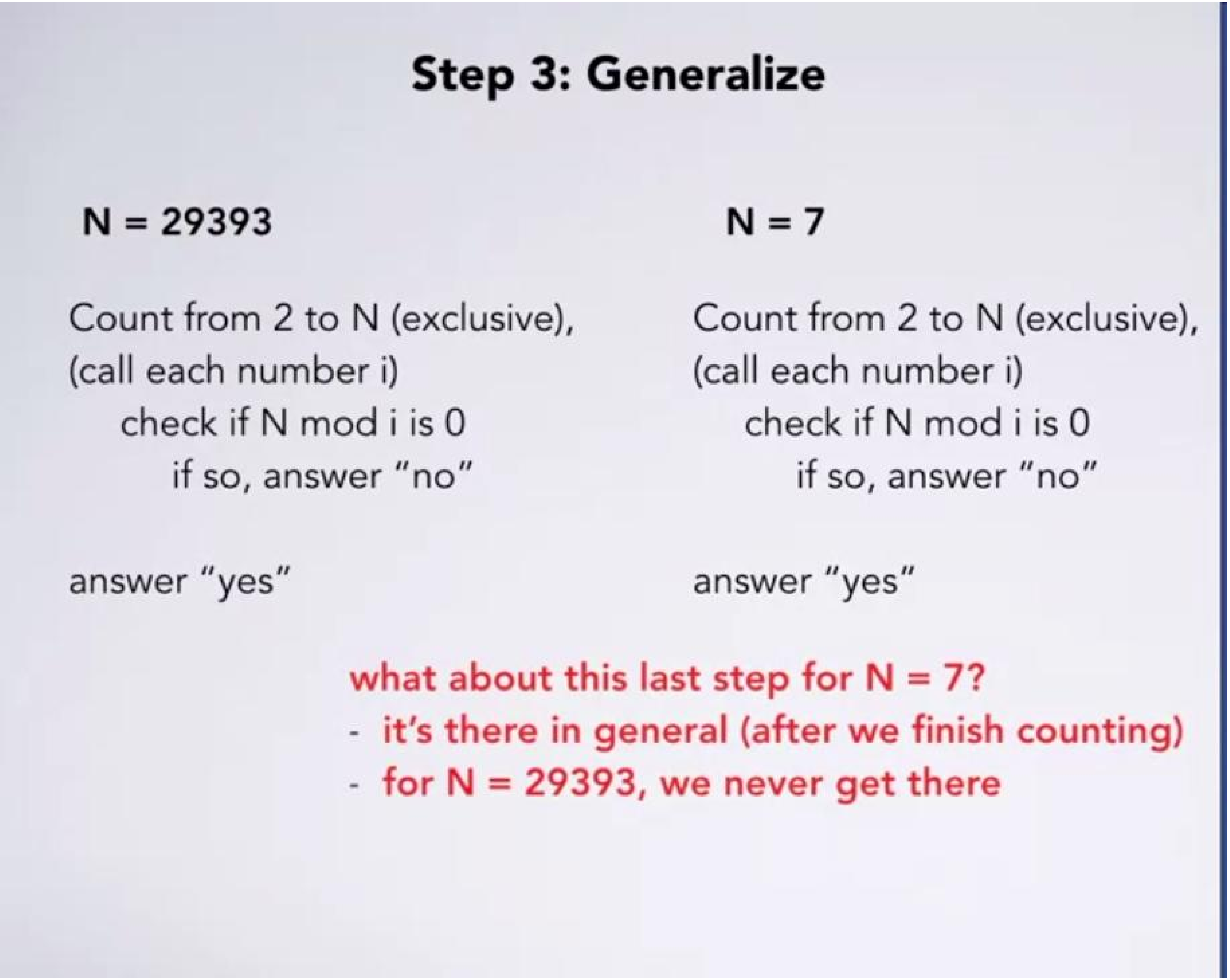
Algorithm:

```
Count from 2 to N (exclusive), (call each number i)
    check if N mod i is 0
        if so, answer “no”
answer “yes”
```

Step 4: Test

Algorithm:

```
Count from 2 to N (exclusive), (call each number i)
    check if N mod i is 0
        if so, answer "no"
answer "yes"
```



Step 3: Generalize

N = 29393

```
Count from 2 to N (exclusive),
(call each number i)
    check if N mod i is 0
        if so, answer "no"

answer "yes"
```

N = 7

```
Count from 2 to N (exclusive),
(call each number i)
    check if N mod i is 0
        if so, answer "no"

answer "yes"
```

what about this last step for N = 7?

- it's there in general (after we finish counting)
- for N = 29393, we never get there

Step 4: Test

Algorithm:

```
Count from 2 to N (exclusive), (call each number i)
    check if N mod i is 0
        if so, answer "no"
answer "yes"
```

May have generalized incorrectly

- Try values you have not used yet

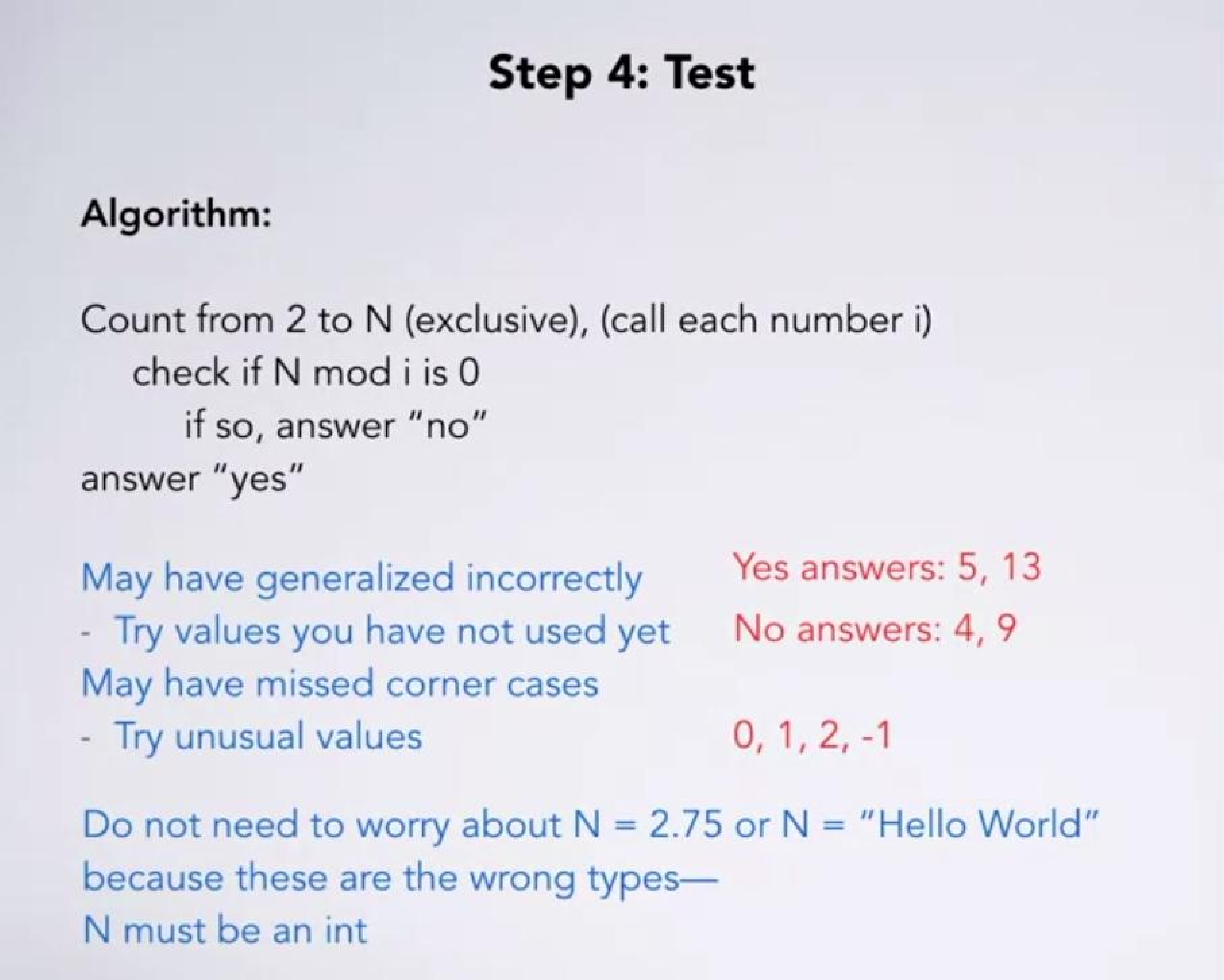
May have missed corner cases

- Try unusual values

Yes answers: 5, 13

No answers: 4, 9

0, 1, 2, -1



Step 4: Test

Algorithm:

```
Count from 2 to N (exclusive), (call each number i)
    check if N mod i is 0
        if so, answer "no"
answer "yes"
```

May have generalized incorrectly

- Try values you have not used yet

Yes answers: 5, 13

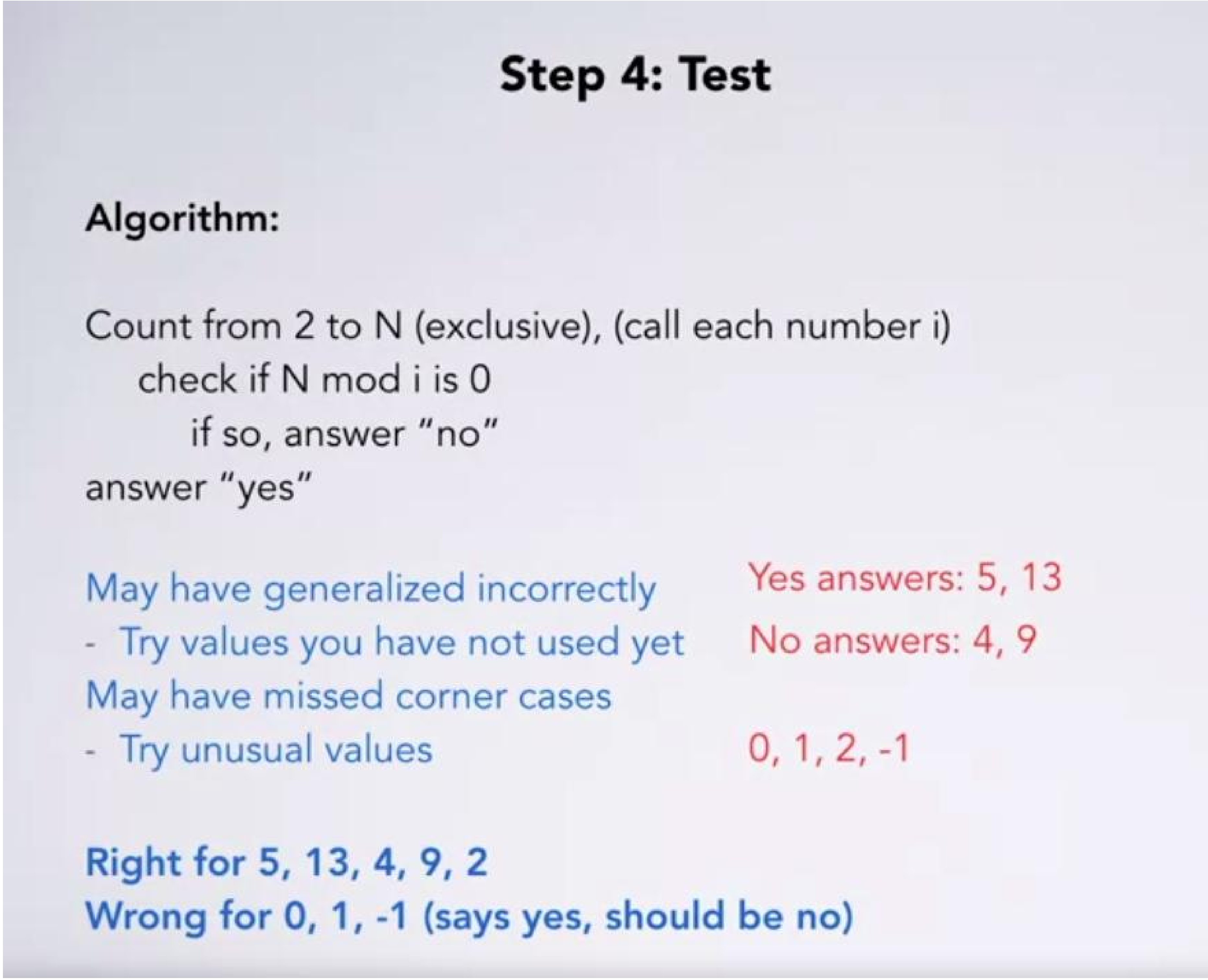
No answers: 4, 9

May have missed corner cases

- Try unusual values

0, 1, 2, -1

Do not need to worry about N = 2.75 or N = "Hello World" because these are the wrong types—N must be an int



Step 4: Test

Algorithm:

```
Count from 2 to N (exclusive), (call each number i)
    check if N mod i is 0
        if so, answer "no"
answer "yes"
```

May have generalized incorrectly

- Try values you have not used yet

Yes answers: 5, 13

No answers: 4, 9

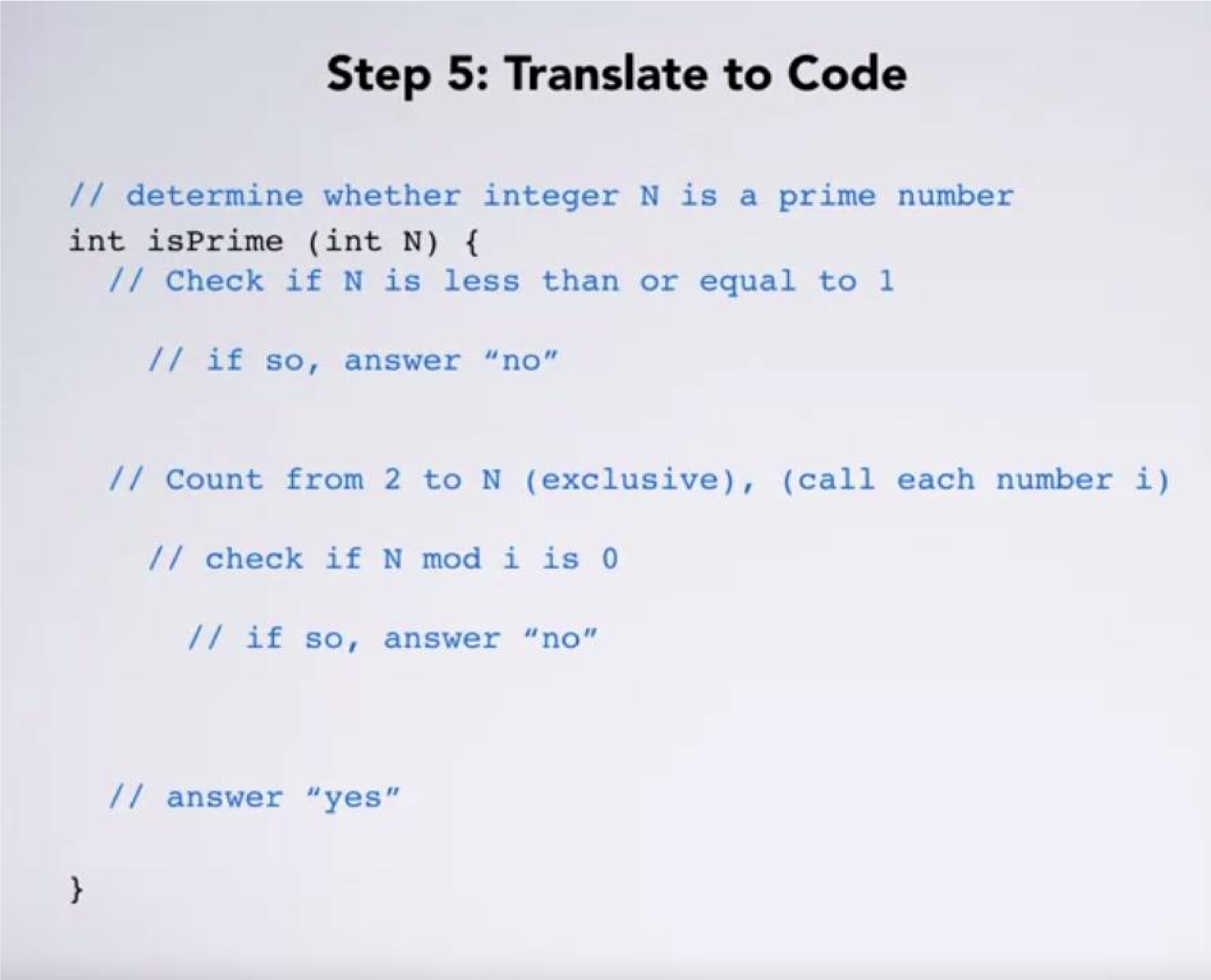
May have missed corner cases

- Try unusual values

0, 1, 2, -1

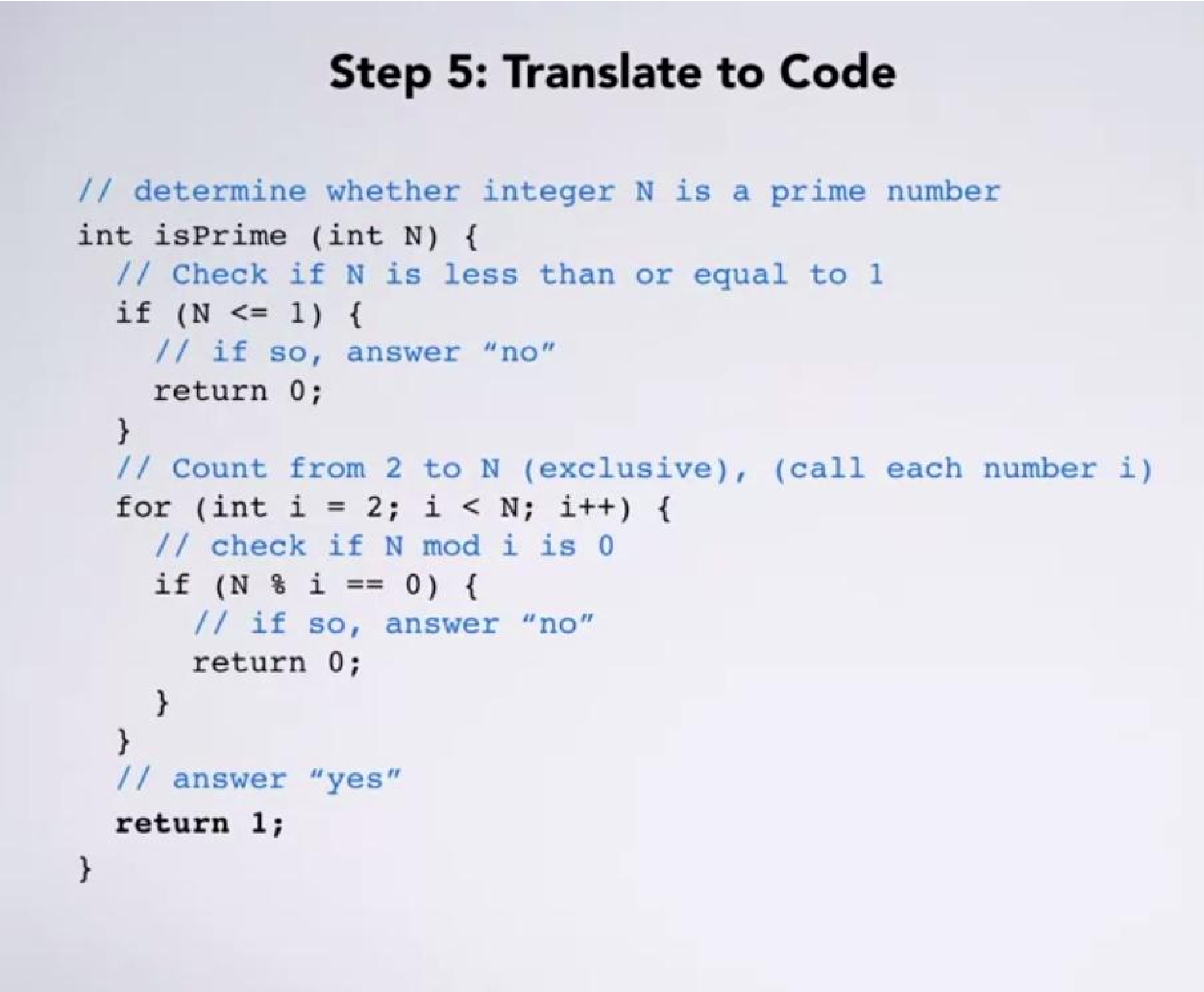
Right for 5, 13, 4, 9, 2

Wrong for 0, 1, -1 (says yes, should be no)



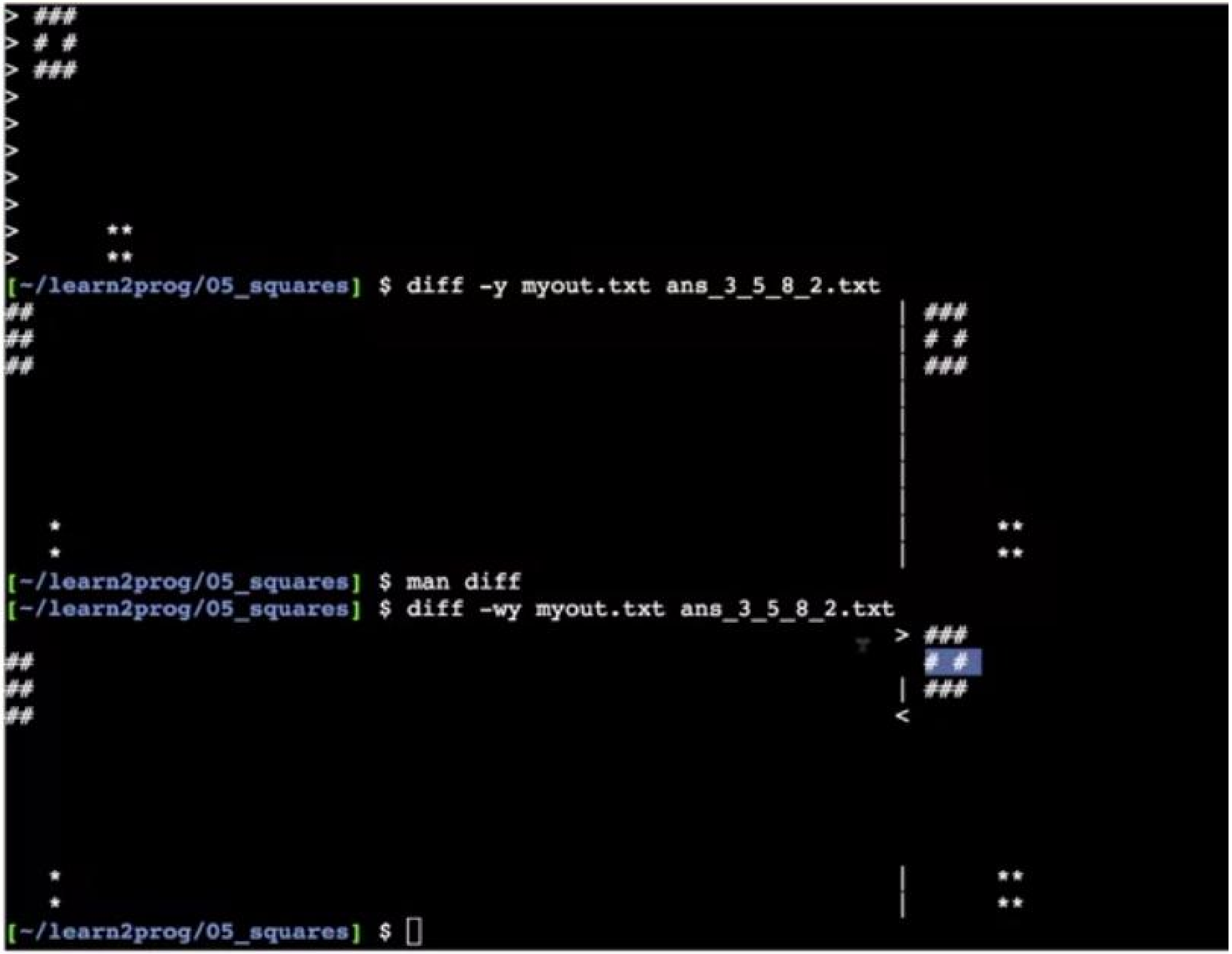
Step 5: Translate to Code

```
// determine whether integer N is a prime number
int isPrime (int N) {
  // Check if N is less than or equal to 1

    // if so, answer "no"


  // Count from 2 to N (exclusive), (call each number i)

    // check if N mod i is 0

      // if so, answer "no"



  // answer "yes"

}
```

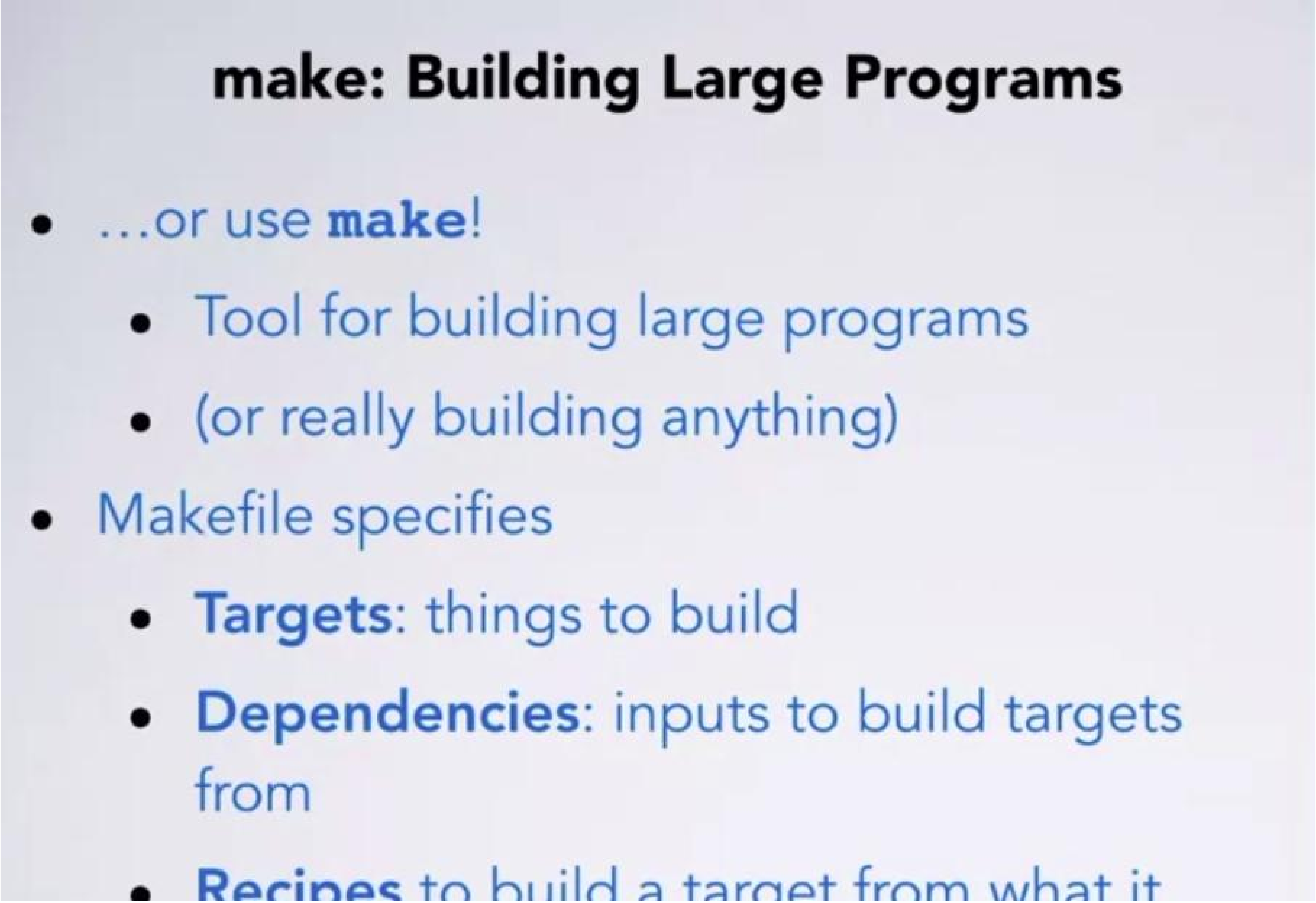


Step 5: Translate to Code

```
// determine whether integer N is a prime number
int isPrime (int N) {
  // Check if N is less than or equal to 1
  if (N <= 1) {
    // if so, answer “no”
    return 0;
  }
  // Count from 2 to N (exclusive), (call each number i)
  for (int i = 2; i < N; i++) {
    // check if N mod i is 0
    if (N % i == 0) {
      // if so, answer “no”
      return 0;
    }
  }
  // answer “yes”
  return 1;
}
```

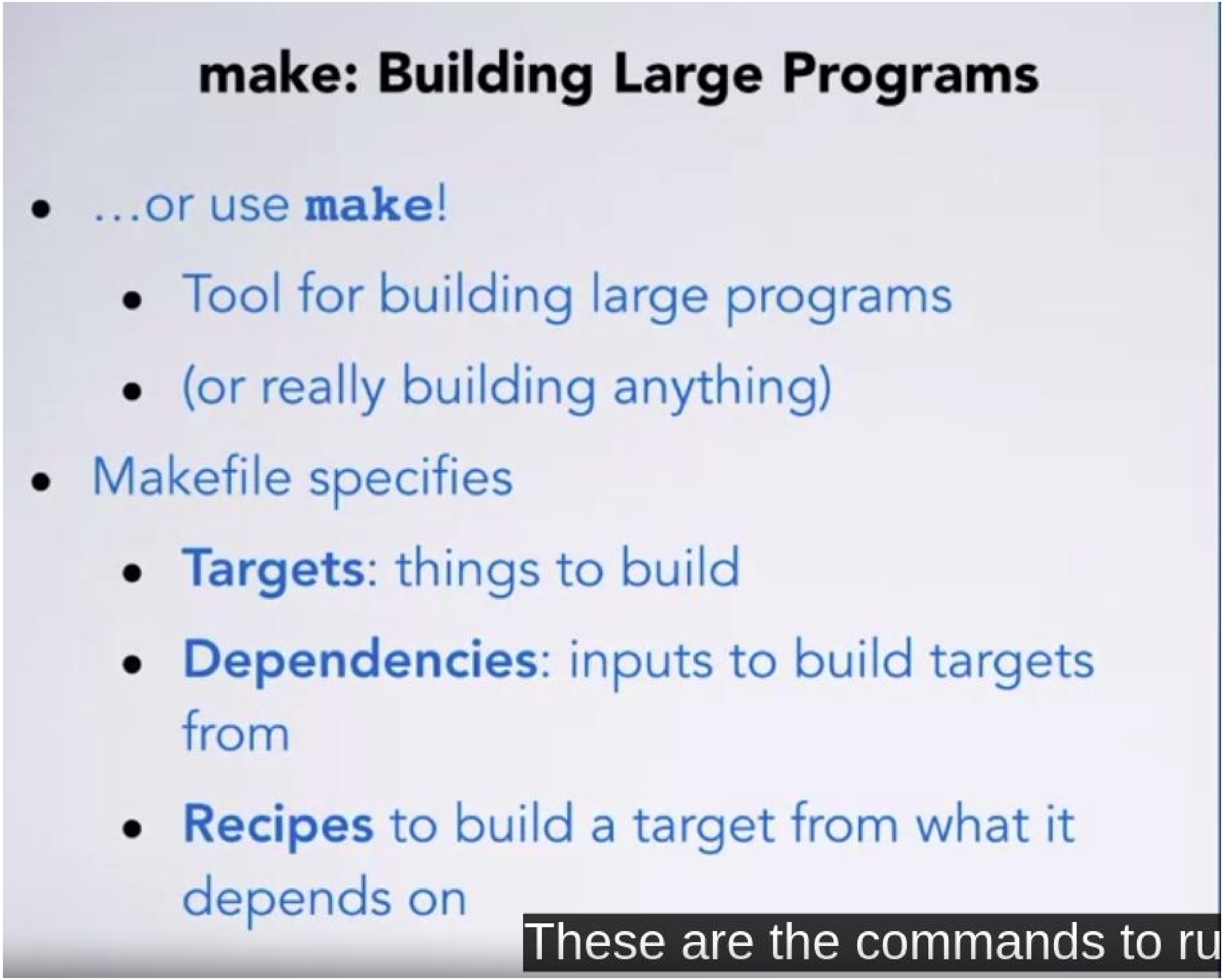


```
> ###
> # #
> ###
>
>
>
>
>
>      **
>      **
[~/learn2prog/05_squares] $ diff -y myout.txt ans_3_5_8_2.txt
##                                                             | ###
##                                                             | # #
##                                                             | ###
                                                               |
                                                               |
                                                               |
                                                               |
                                                               |
   *                                                           |      **
   *                                                           |      **
[~/learn2prog/05_squares] $ man diff
[~/learn2prog/05_squares] $ diff -wy myout.txt ans_3_5_8_2.txt
                                                               > ###
##                                                               # #
##                                                             | ###
##                                                             <





   *                                                           |      **
   *                                                           |      **
[~/learn2prog/05_squares] $ 
```



make: Building Large Programs

- …or use **`make`**!
  - Tool for building large programs
  - (or really building anything)
- Makefile specifies
  - **Targets**: things to build
  - **Dependencies**: inputs to build targets from
  - **Recipes** to build a target from what it

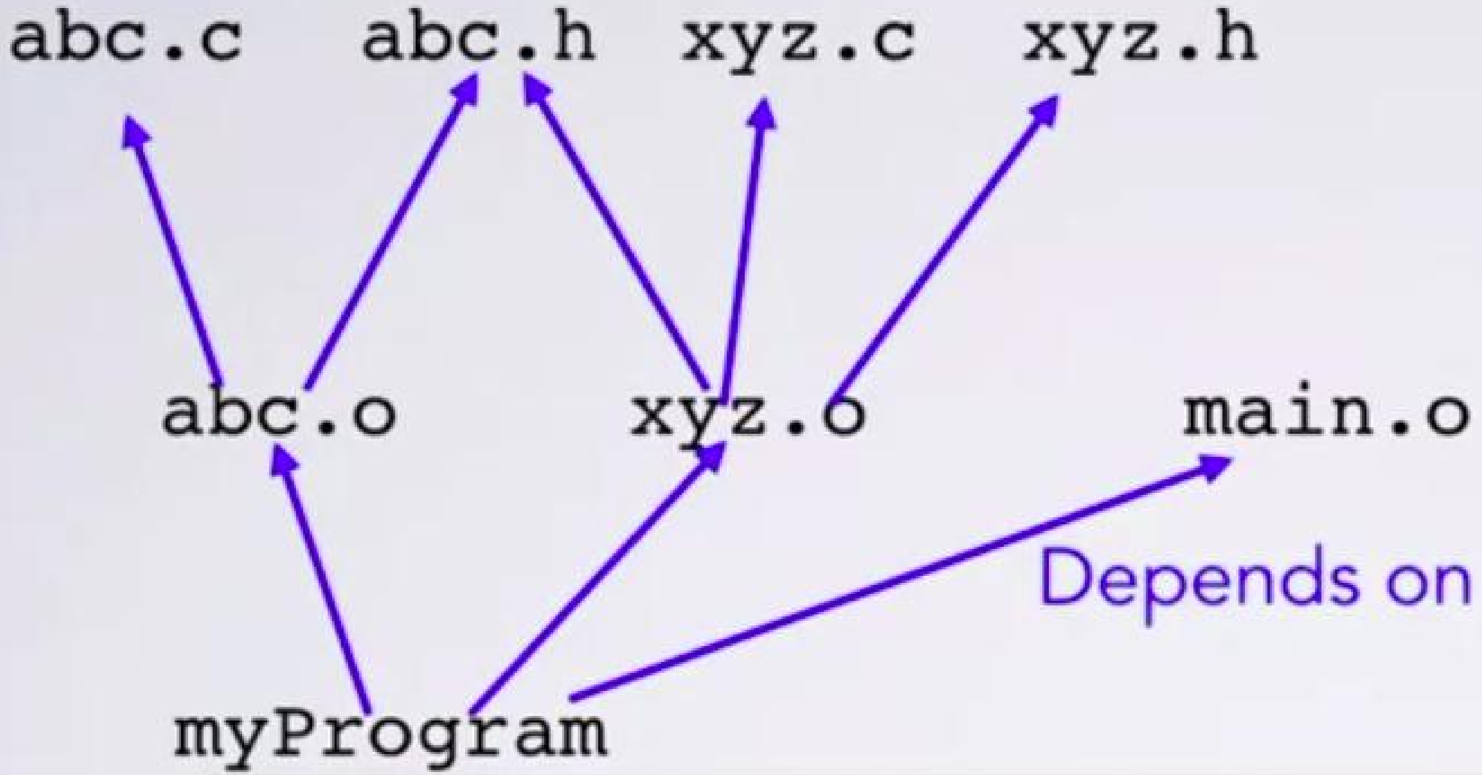


make: Building Large Programs

- …or use **`make`**!
  - Tool for building large programs
  - (or really building anything)
- Makefile specifies
  - **Targets**: things to build
  - **Dependencies**: inputs to build targets from
  - **Recipes** to build a target from what it depends on

These are the commands to ru

Dependencies



abc.c abc.h xyz.c xyz.h

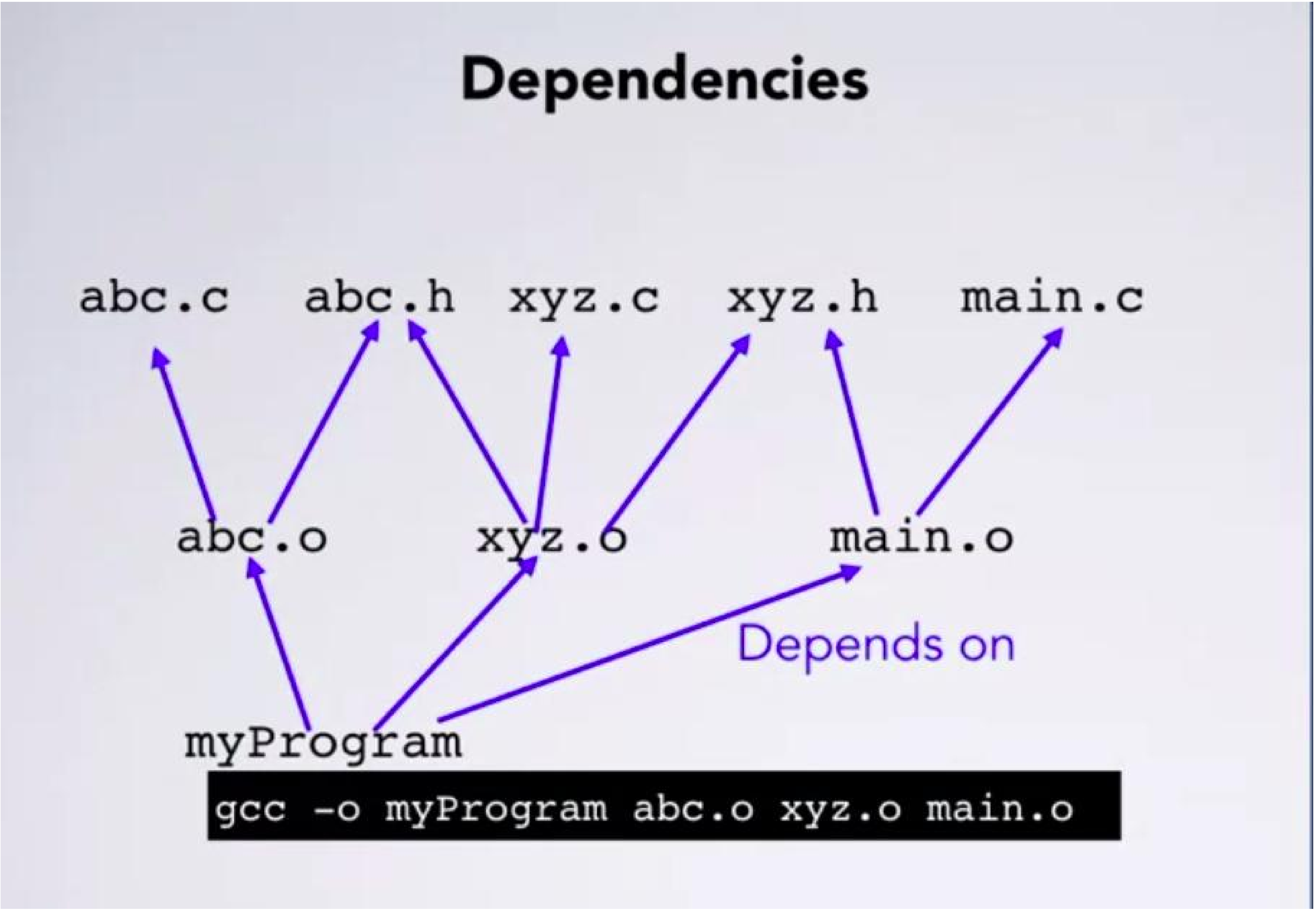
abc.o xyz.o main.o

Depends on

myProgram

```
gcc -o myProgram abc.o xyz.o main.o
```

on a C file and



Dependencies

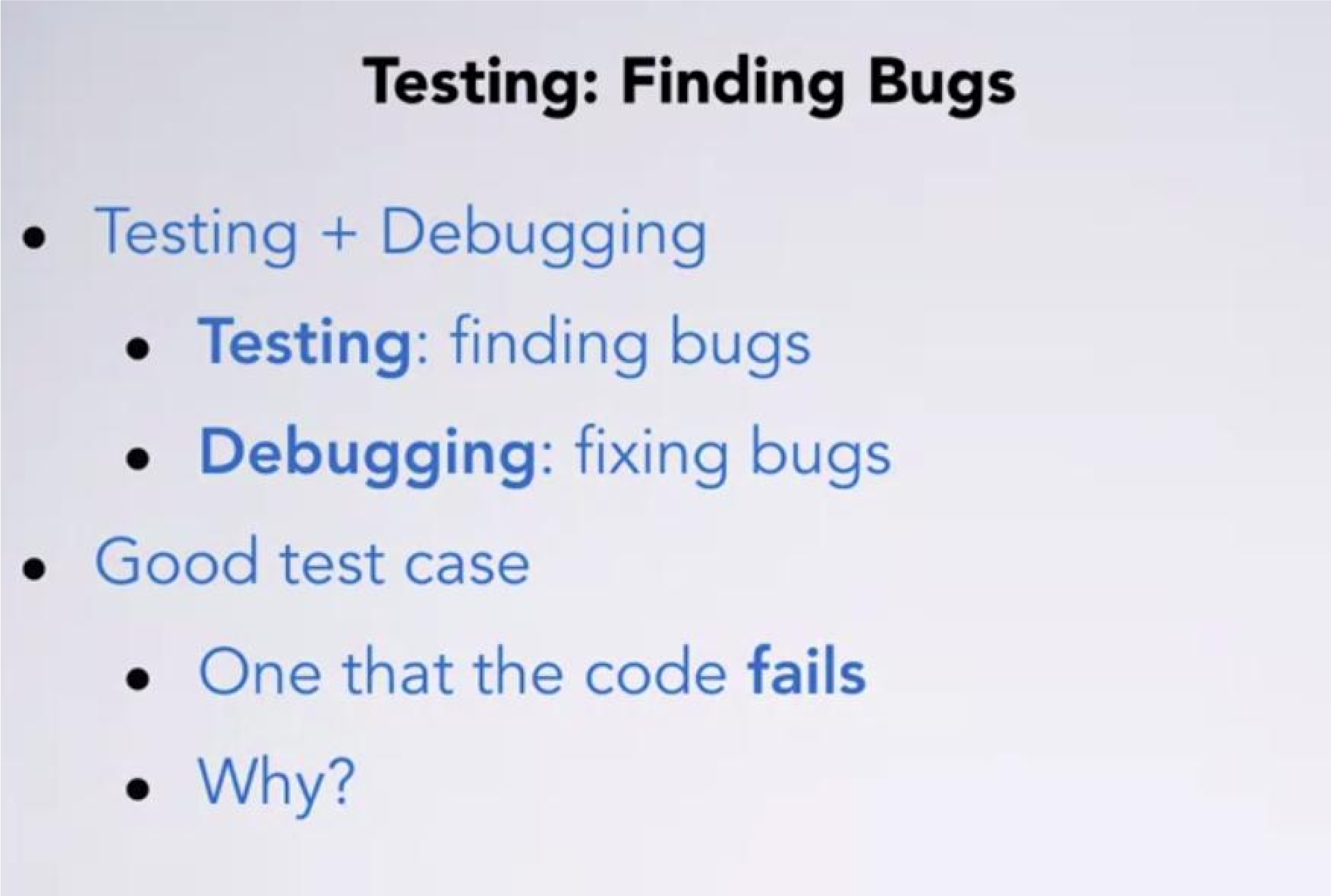
abc.c abc.h xyz.c xyz.h main.c

abc.o xyz.o main.o

Depends on

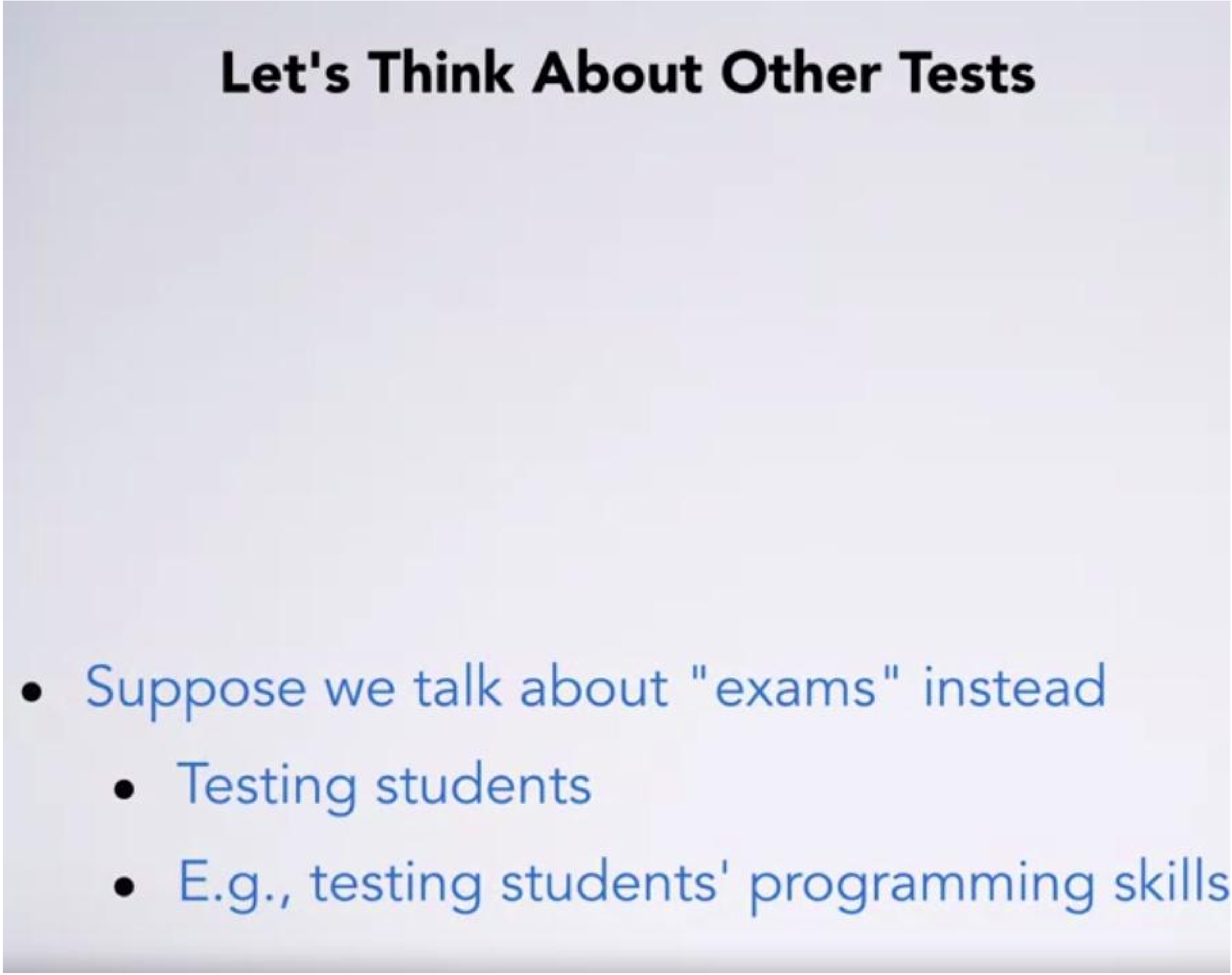
myProgram

```
gcc -o myProgram abc.o xyz.o main.o
```



Testing: Finding Bugs

- Testing + Debugging
  - **Testing**: finding bugs
  - **Debugging**: fixing bugs
- Good test case
  - One that the code **fails**
  - Why?



Let's Think About Other Tests

- Suppose we talk about "exams" instead
  - Testing students
  - E.g., testing students' programming skills

Let's Think About Other Tests

Question 1:

Choose the picture of a computer:

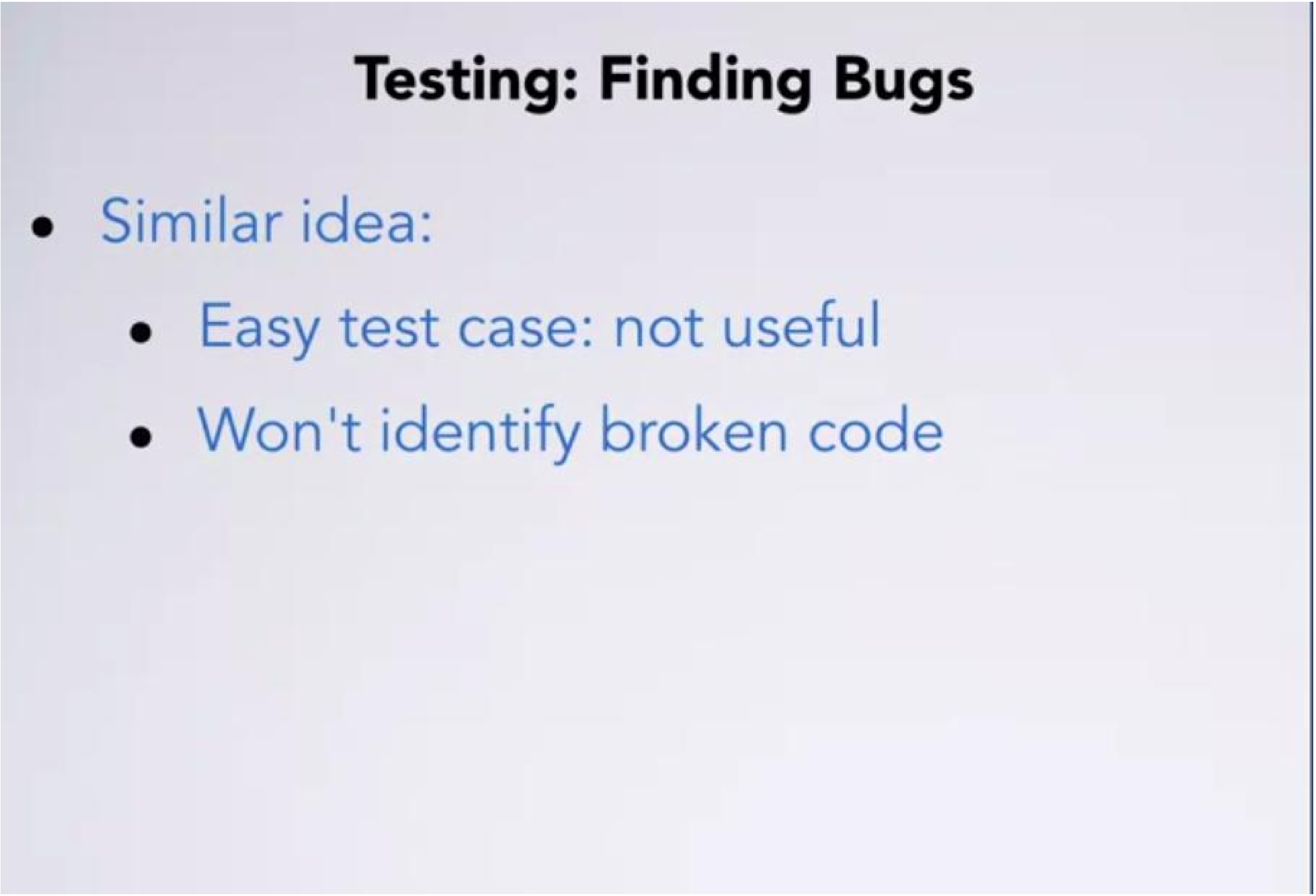
(a)

(b)

(c)

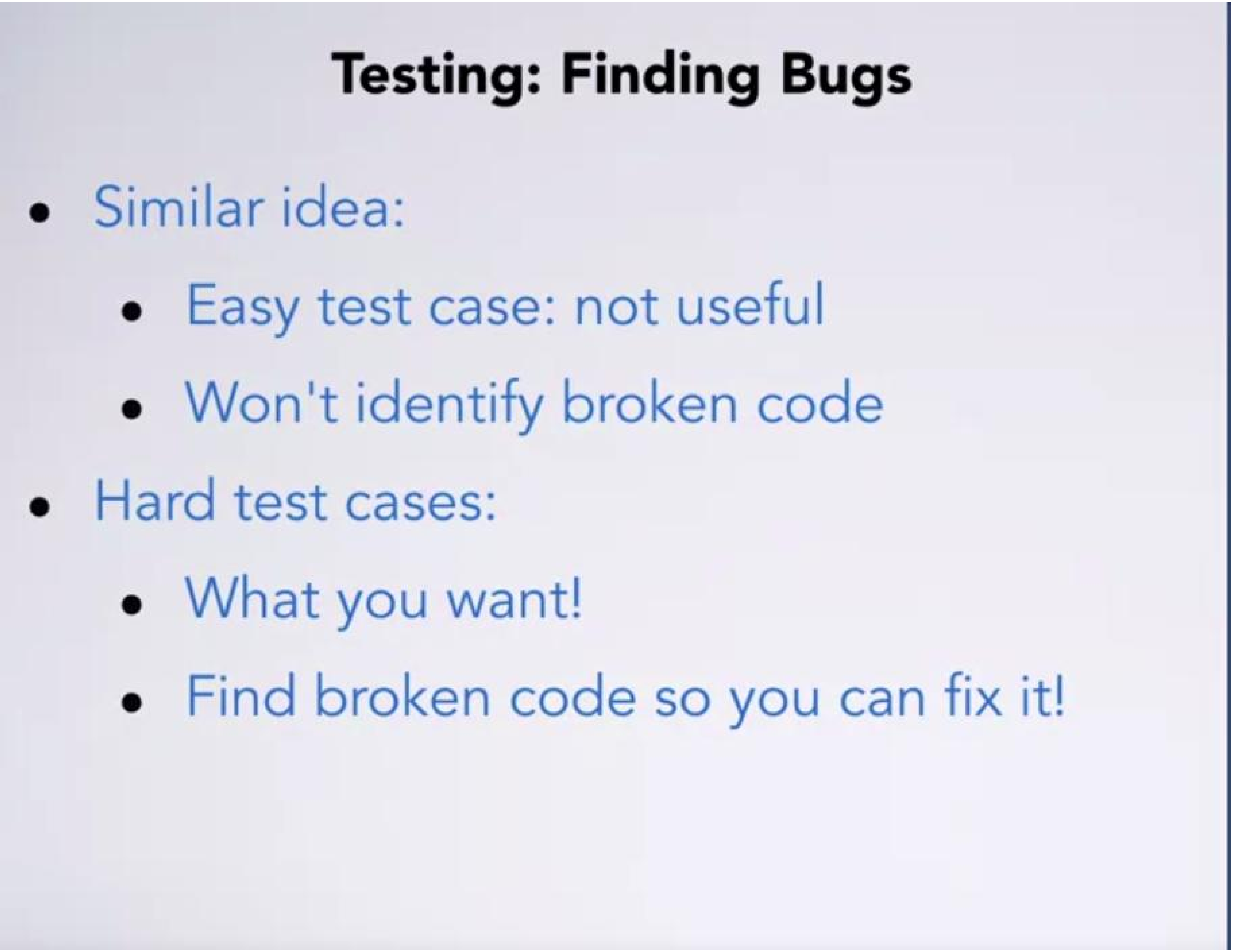
(d)

- Nobody would get this wrong
  - Even with no programming knowledge
  - Too easy



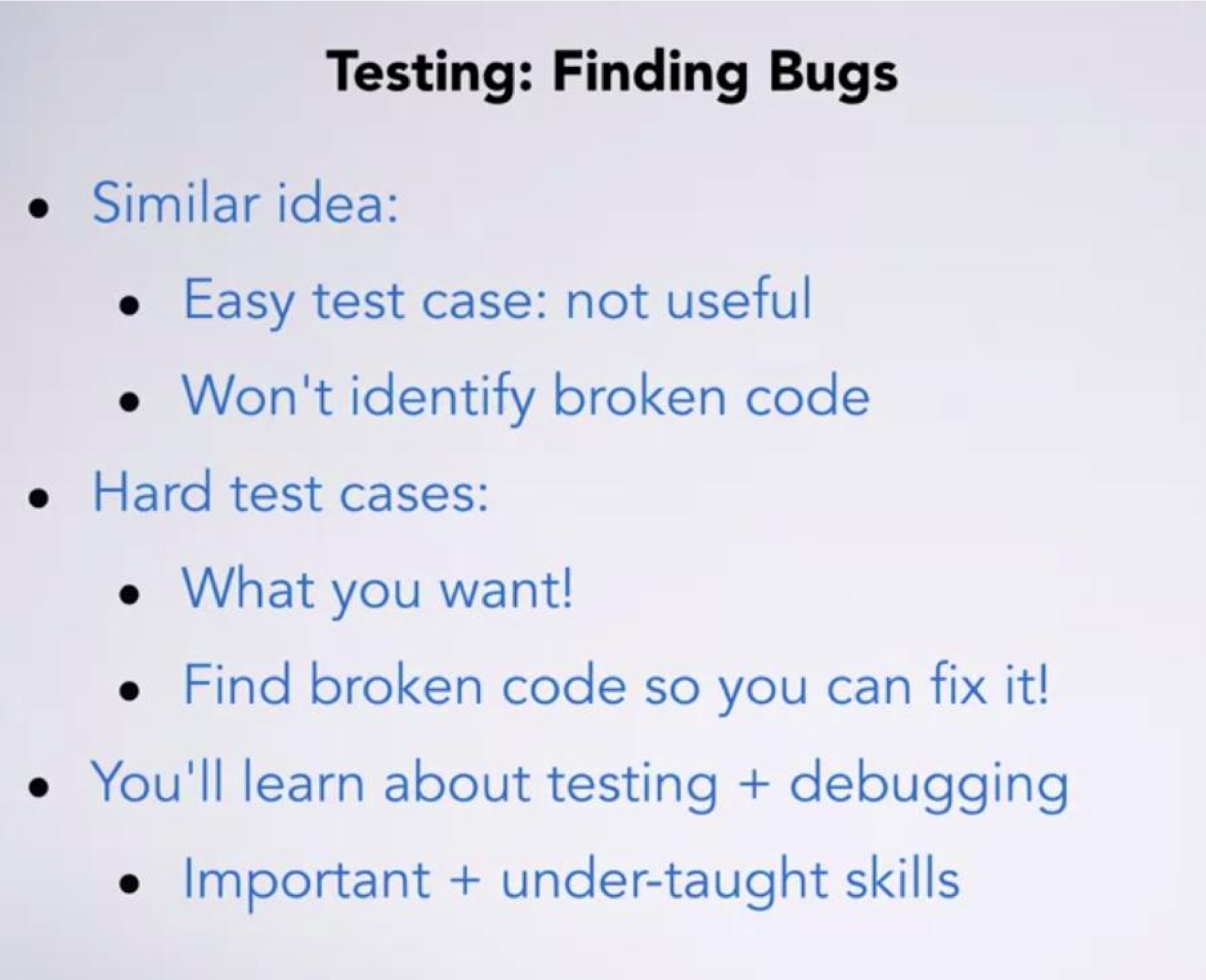
Testing: Finding Bugs

- Similar idea:
    - Easy test case: not useful
    - Won't identify broken code



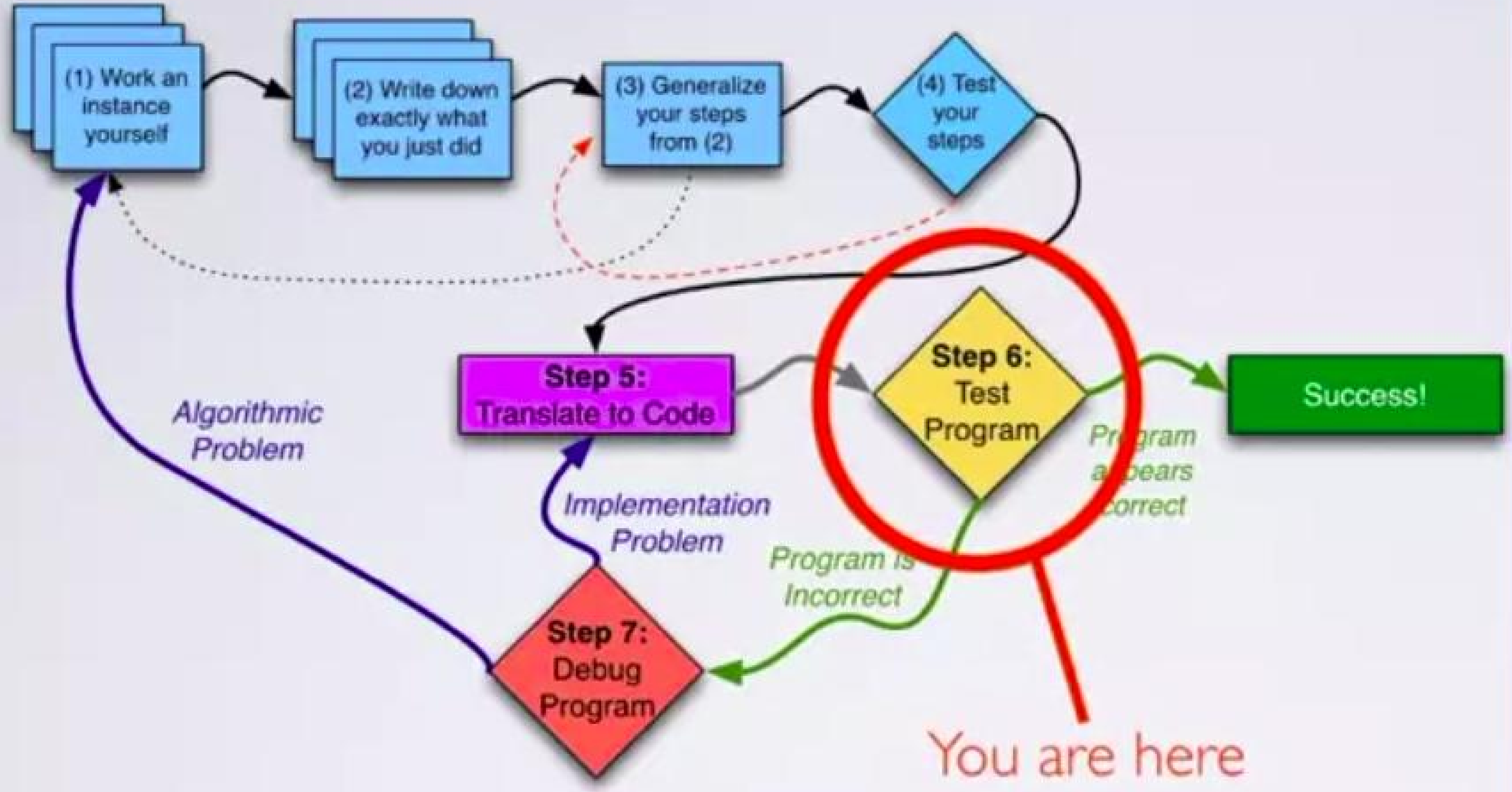
Testing: Finding Bugs

- Similar idea:
  - Easy test case: not useful
  - Won't identify broken code
- Hard test cases:
  - What you want!
  - Find broken code so you can fix it!



Testing: Finding Bugs

- Similar idea:
  - Easy test case: not useful
  - Won't identify broken code
- Hard test cases:
  - What you want!
  - Find broken code so you can fix it!
- You'll learn about testing + debugging
  - Important + under-taught skills



(1) Work an instance yourself

(2) Write down exactly what you just did

(3) Generalize your steps from (2)

(4) Test your steps

Step 5: Translate to Code

Step 6: Test Program

Success!

Program appears correct

Program is Incorrect

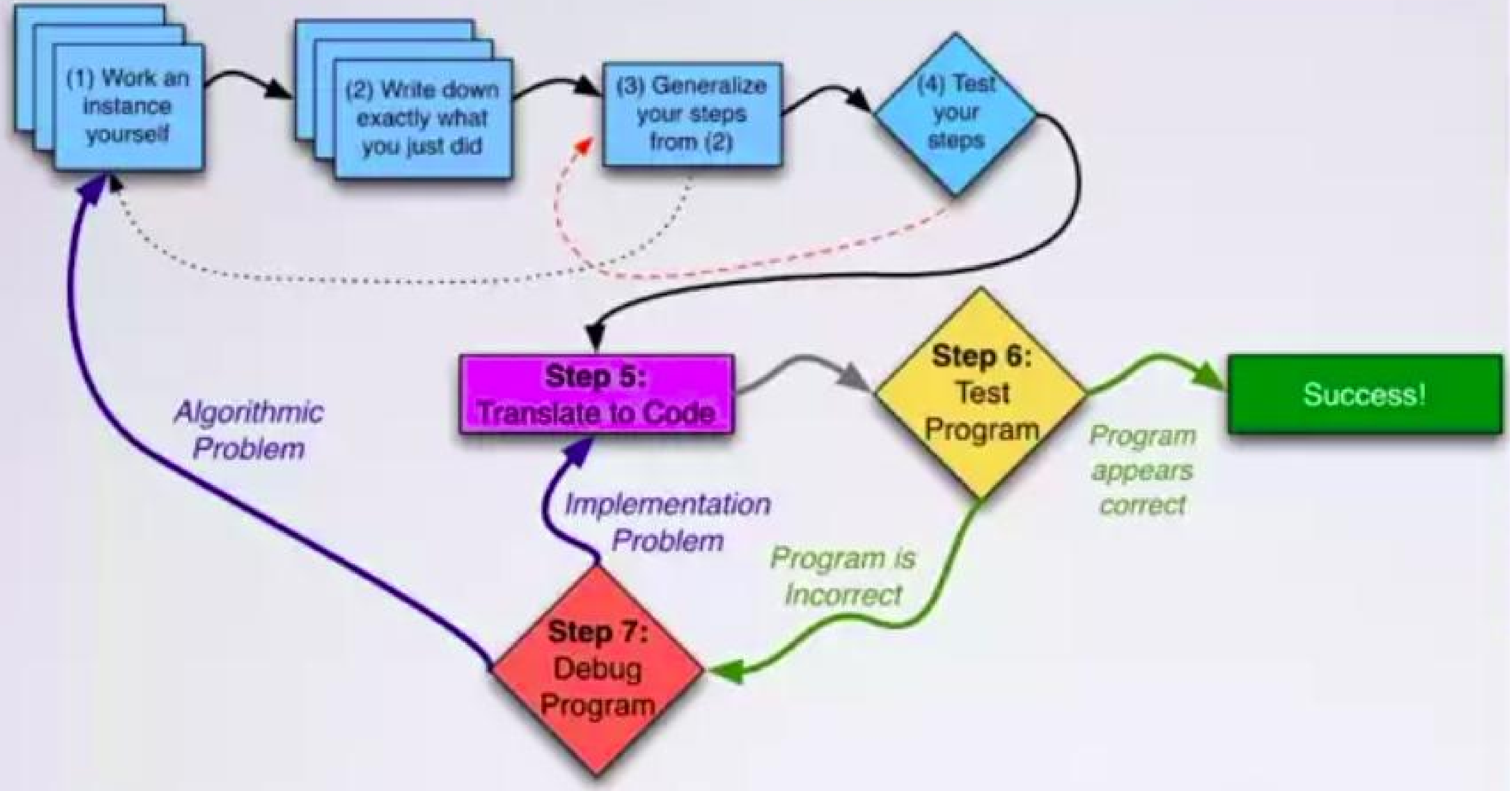
Step 7: Debug Program

Implementation Problem

Algorithmic Problem

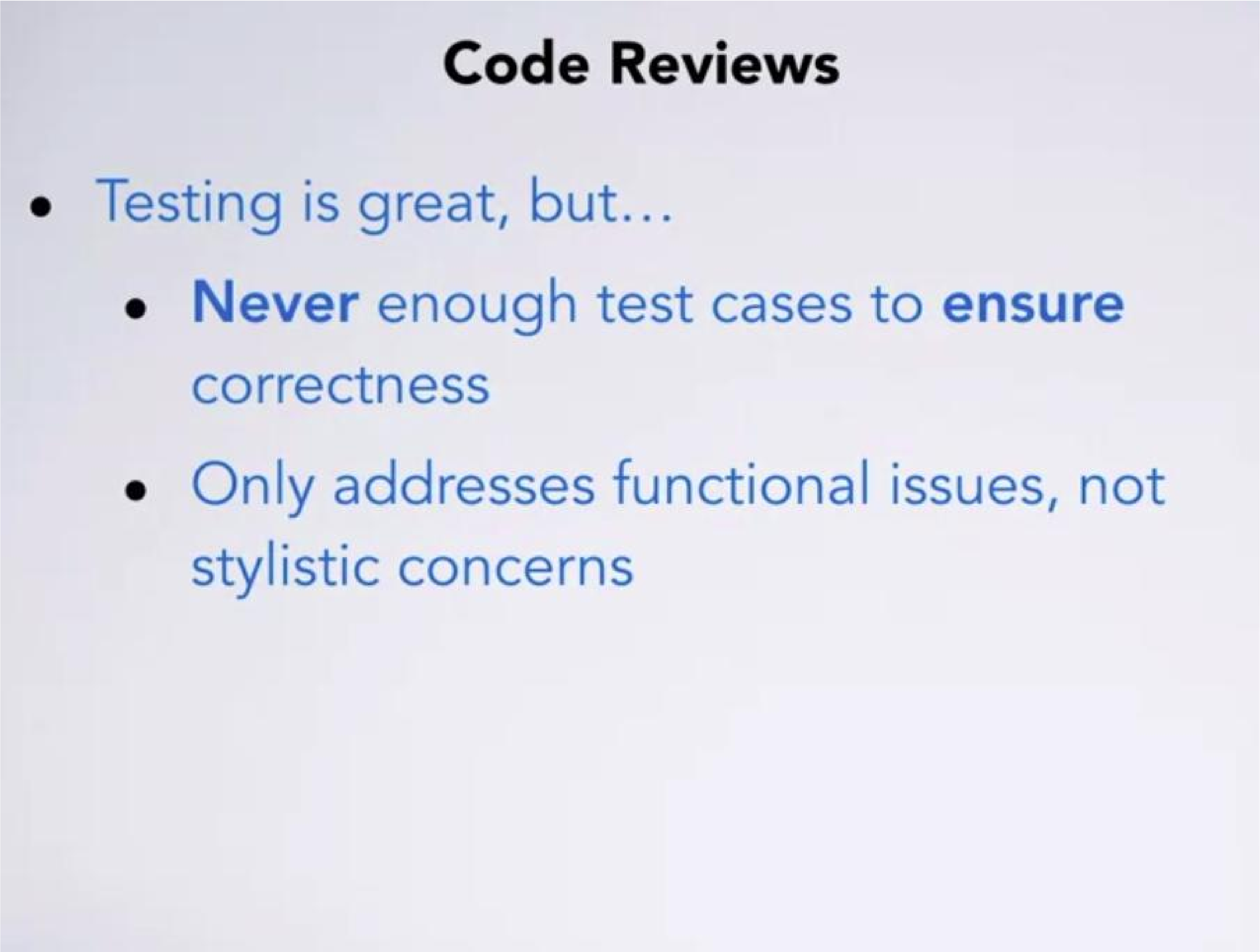
You are here

Test Driven Development



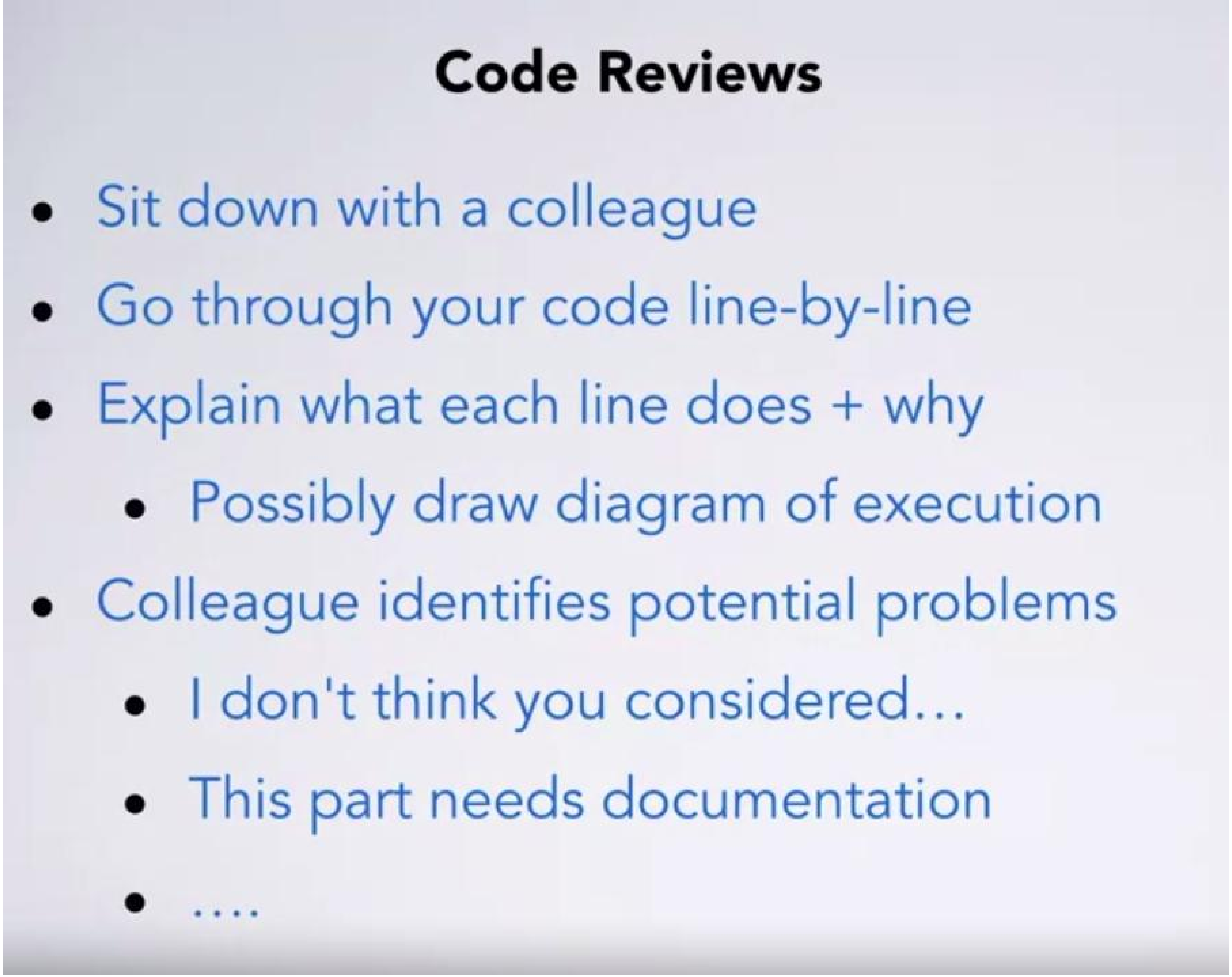
Come up with test cases **first** (matches nicely with step 1)

Test cases already ready when you get to Step 6



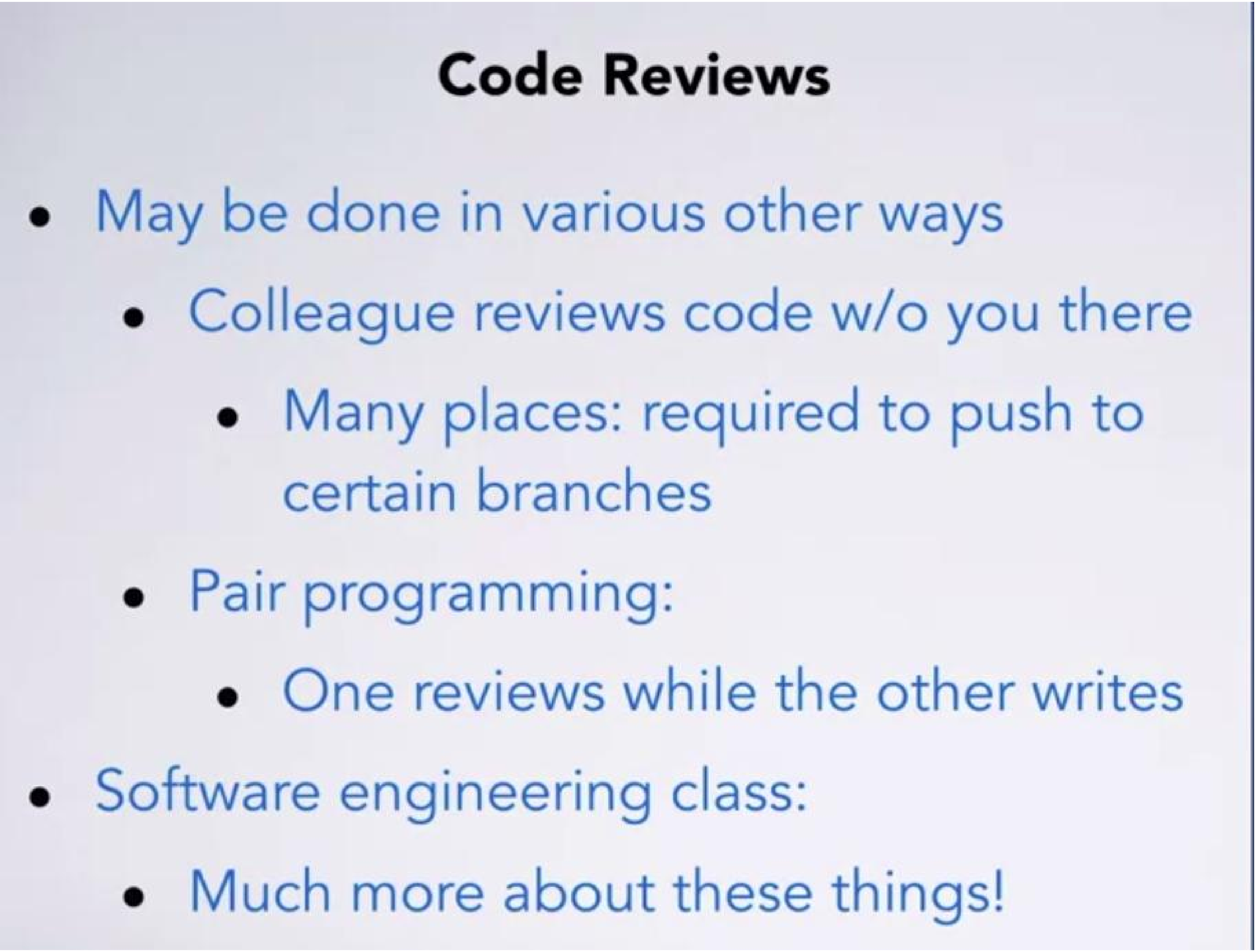
Code Reviews

- Testing is great, but…
  - **Never** enough test cases to **ensure** correctness
  - Only addresses functional issues, not stylistic concerns



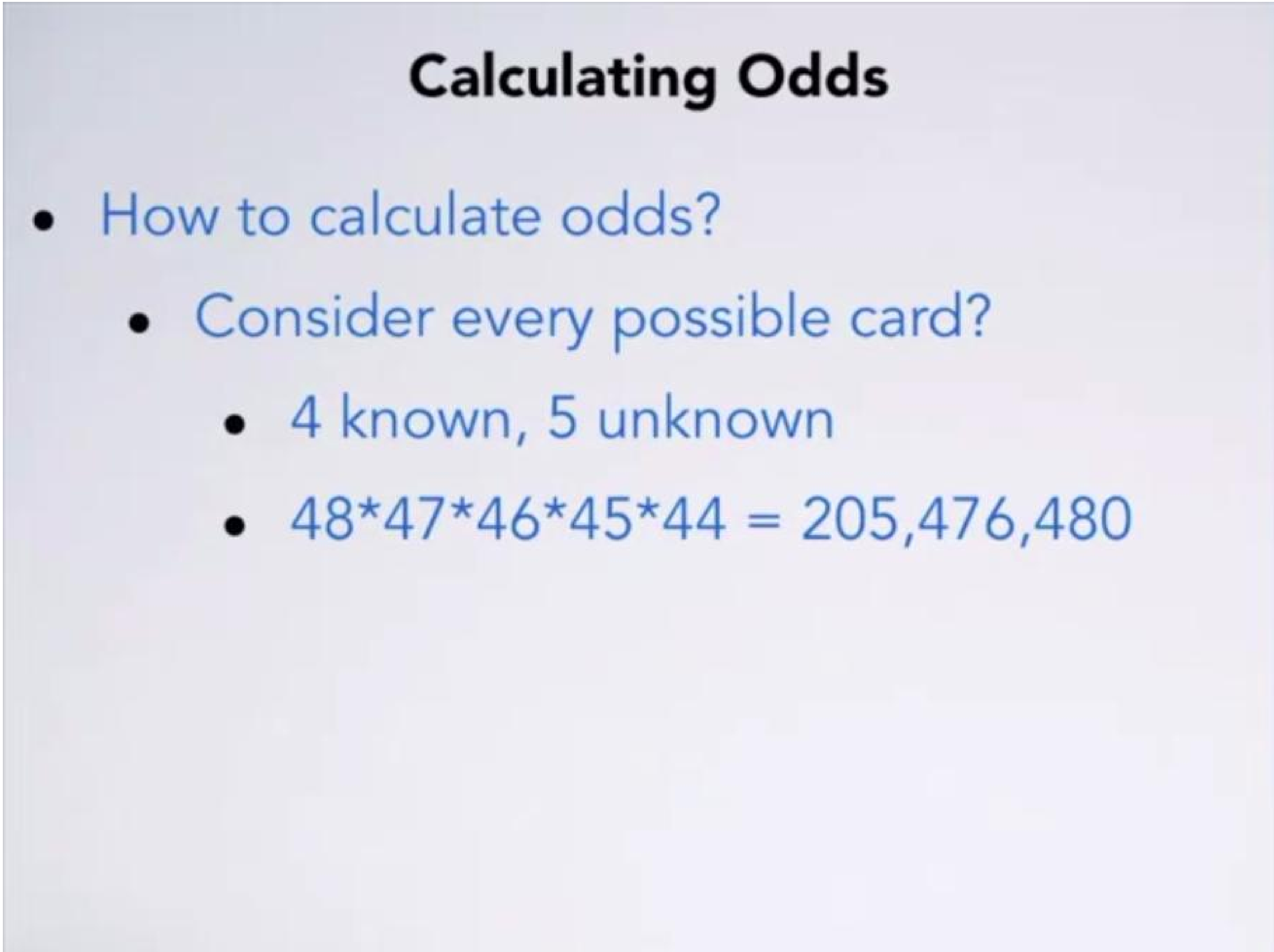
Code Reviews

- Sit down with a colleague
- Go through your code line-by-line
- Explain what each line does + why
  - Possibly draw diagram of execution
- Colleague identifies potential problems
  - I don't think you considered…
  - This part needs documentation
  - ….



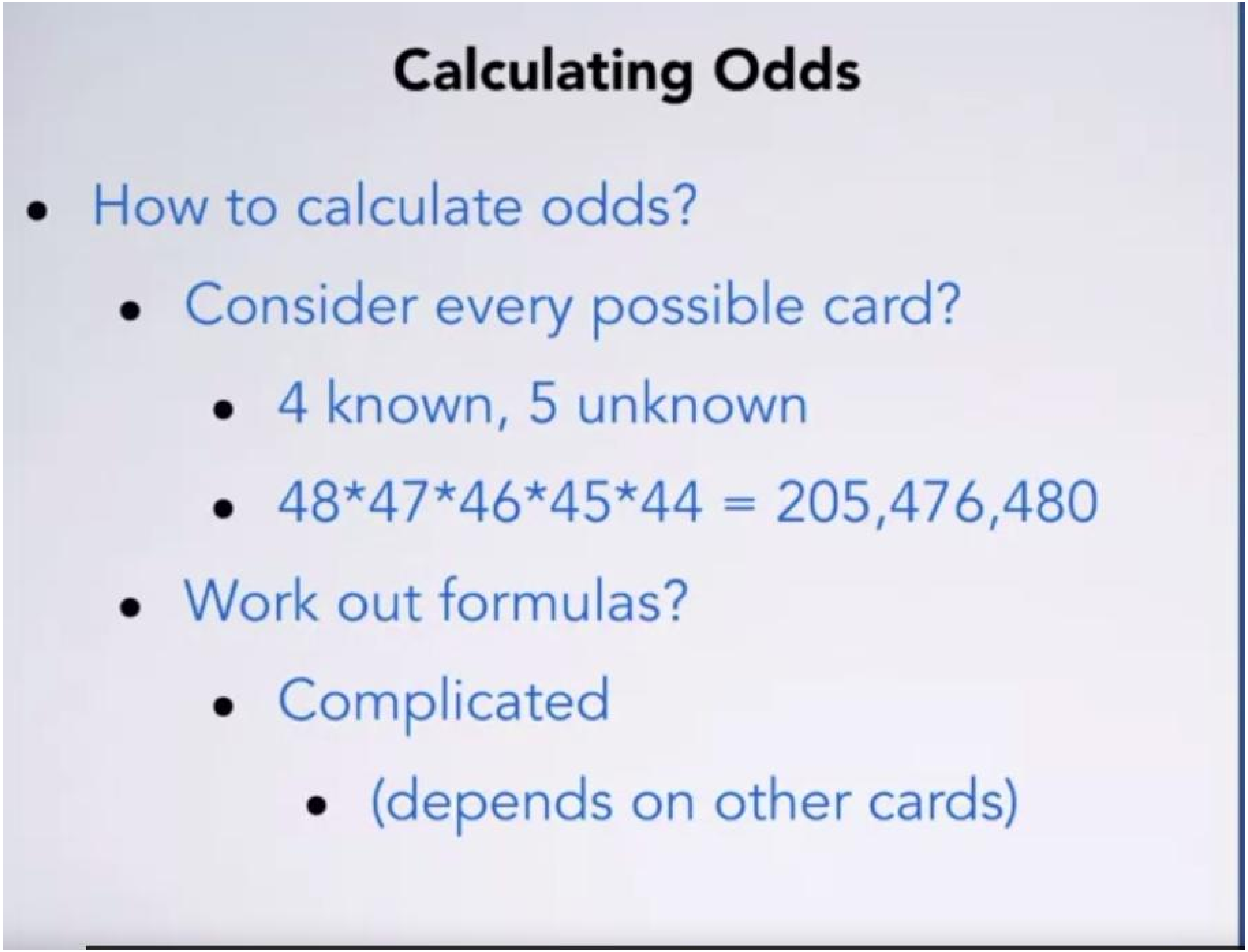
Code Reviews

- May be done in various other ways
  - Colleague reviews code w/o you there
    - Many places: required to push to certain branches
  - Pair programming:
    - One reviews while the other writes
- Software engineering class:
  - Much more about these things!



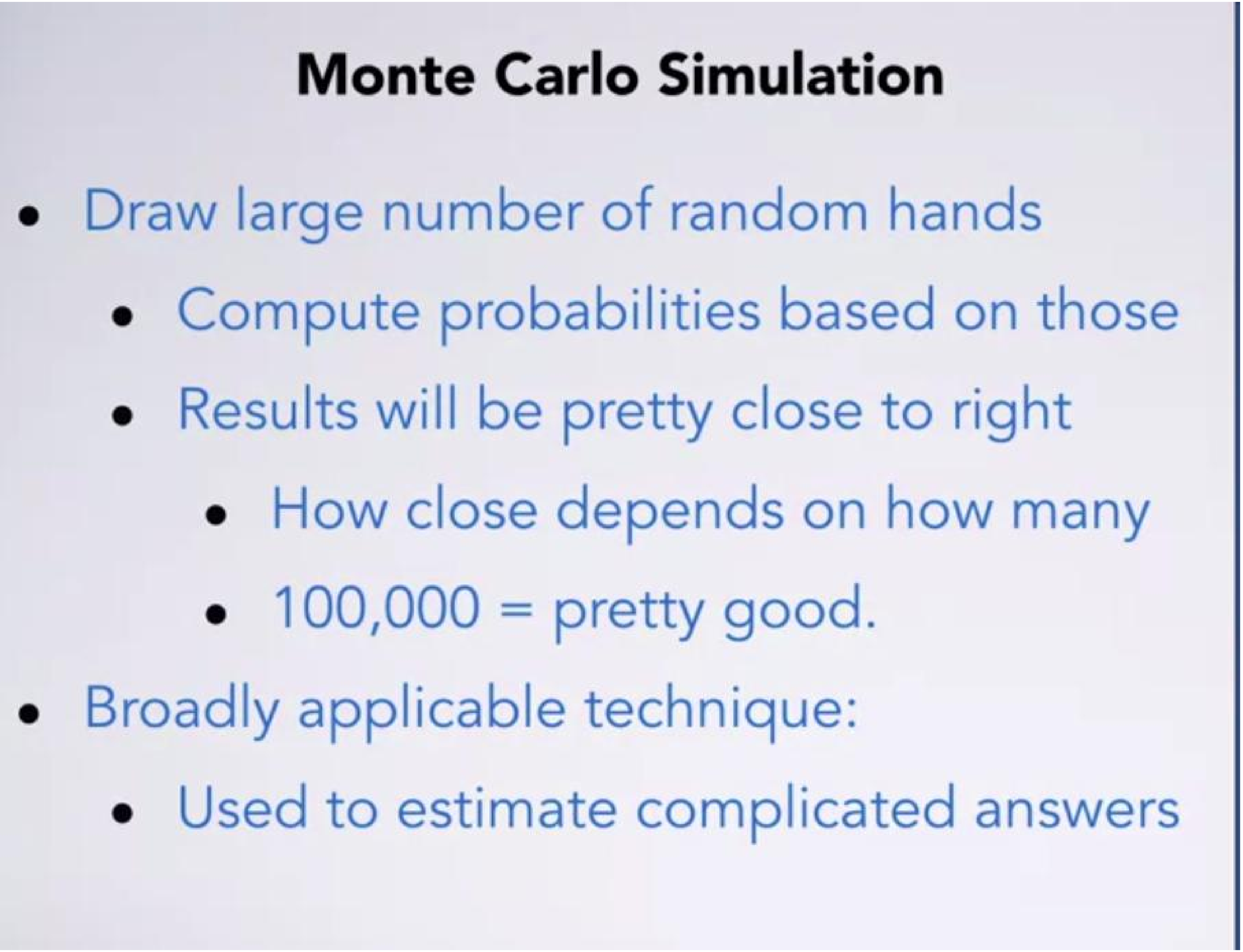
Calculating Odds

- How to calculate odds?
  - Consider every possible card?
    - 4 known, 5 unknown
    - 48*47*46*45*44 = 205,476,480



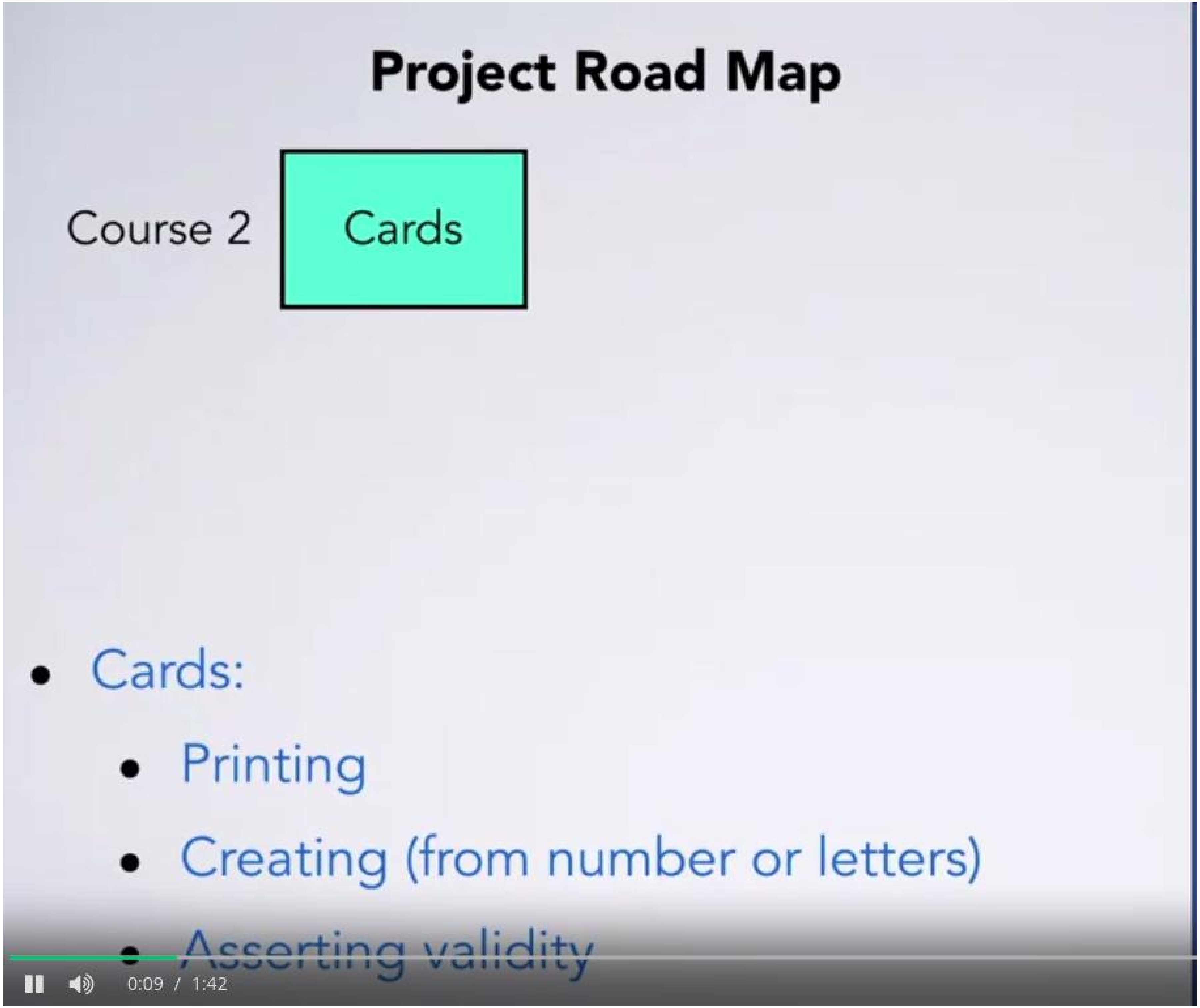
Calculating Odds

- How to calculate odds?
  - Consider every possible card?
    - 4 known, 5 unknown
    - 48*47*46*45*44 = 205,476,480
  - Work out formulas?
    - Complicated
      - (depends on other cards)



Monte Carlo Simulation

- Draw large number of random hands
  - Compute probabilities based on those
  - Results will be pretty close to right
    - How close depends on how many
    - 100,000 = pretty good.
- Broadly applicable technique:
  - Used to estimate complicated answers

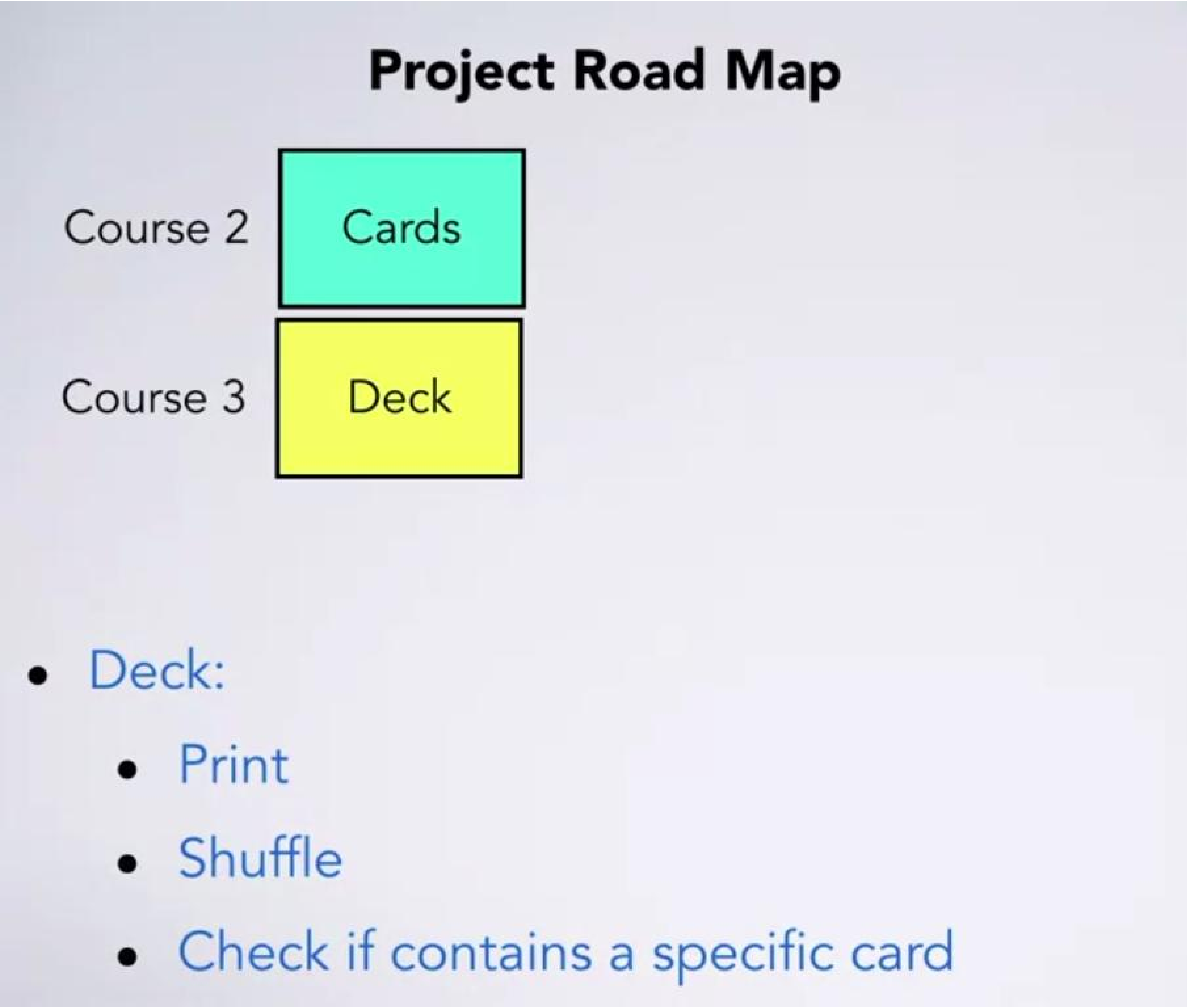


Project Road Map

Course 2

Cards

- Cards:
    - Printing
    - Creating (from number or letters)
    - Asserting validity

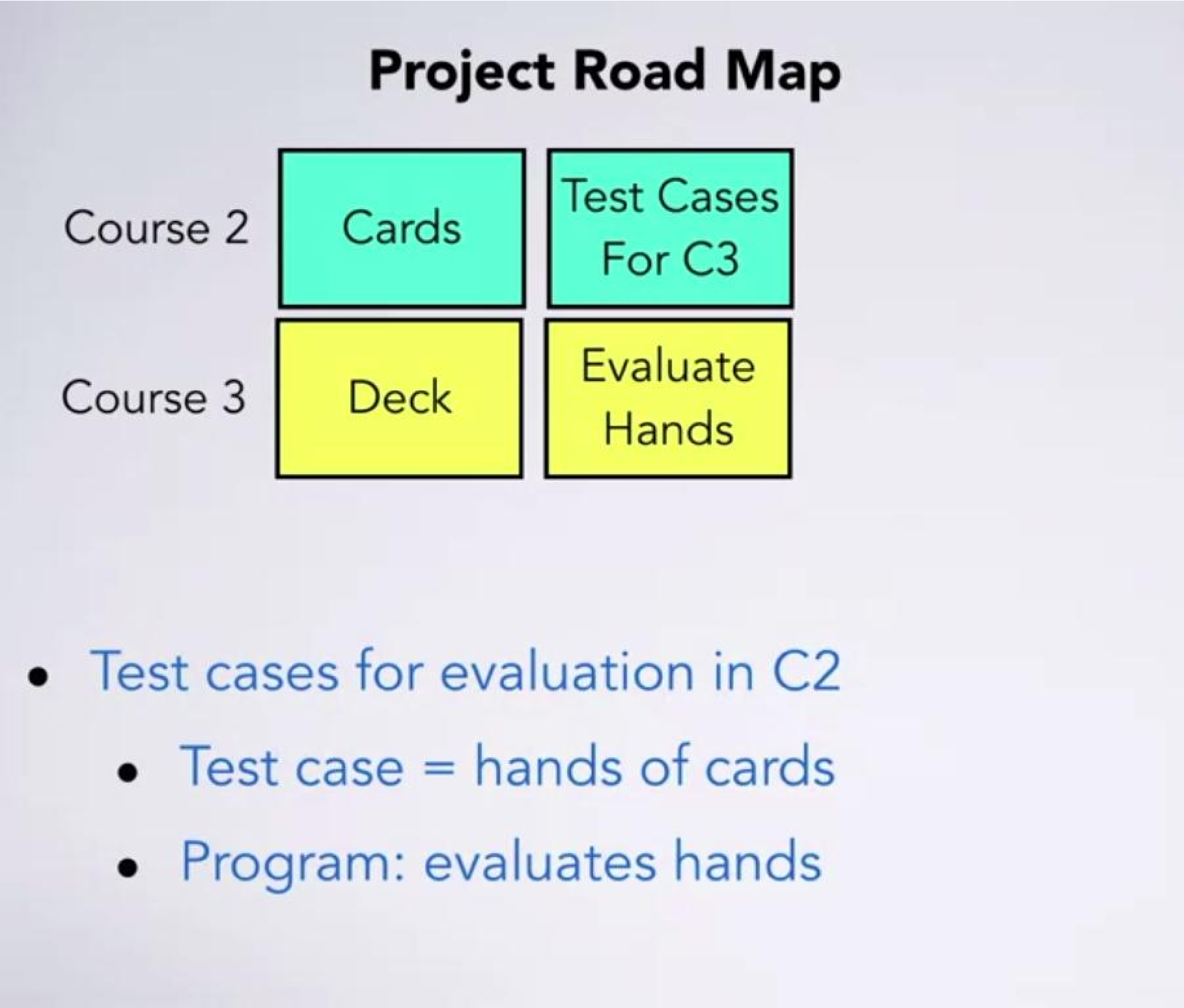


Project Road Map

Course 2 Cards

Course 3 Deck

- Deck:
  - Print
  - Shuffle
  - Check if contains a specific card

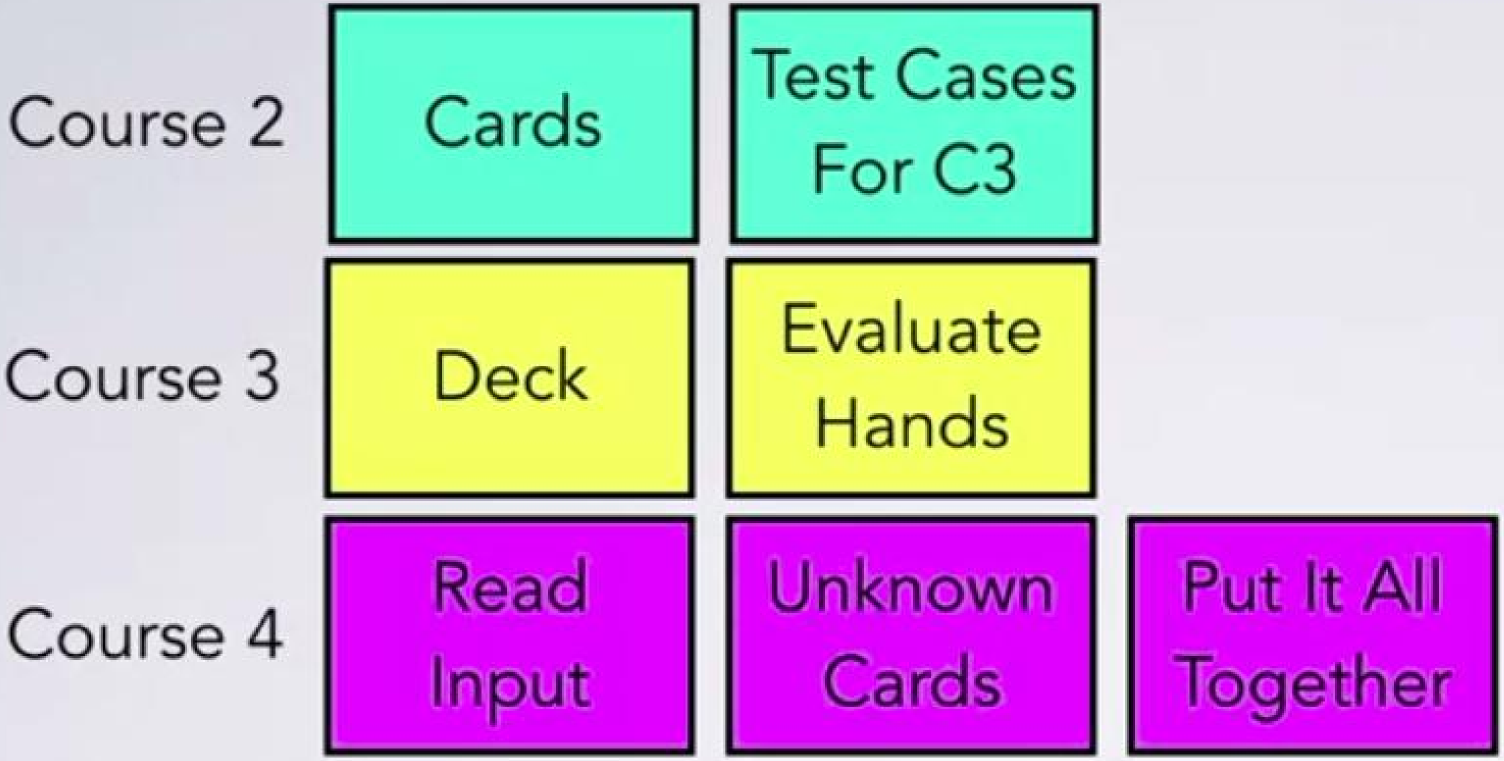


Project Road Map

Course 2	Cards	Test Cases For C3
Course 3	Deck	Evaluate Hands

- Test cases for evaluation in C2
  - Test case = hands of cards
  - Program: evaluates hands

Project Road Map



Course 2	Cards	Test Cases For C3	
Course 3	Deck	Evaluate Hands	
Course 4	Read Input	Unknown Cards	Put It All Together

- Read input
- Handle unknown (?0, ?1) cards
- Put it all together: main