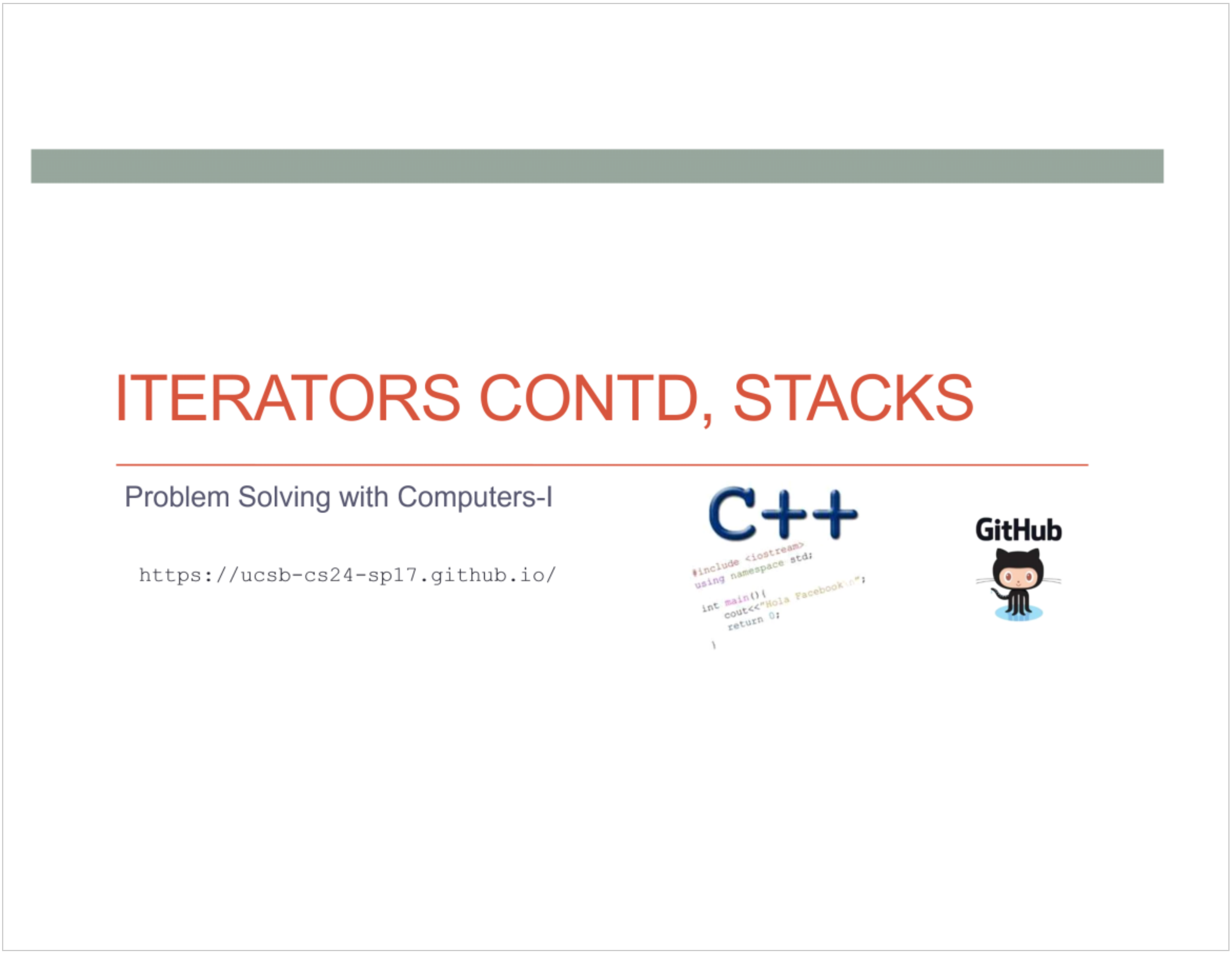
Lecture 12

Monday, May 15, 2017 12:30 PM



ITERATORS CONTD, STACKS

Problem Solving with Computers-I

https://ucsb-cs24-sp17.github.io/

C++

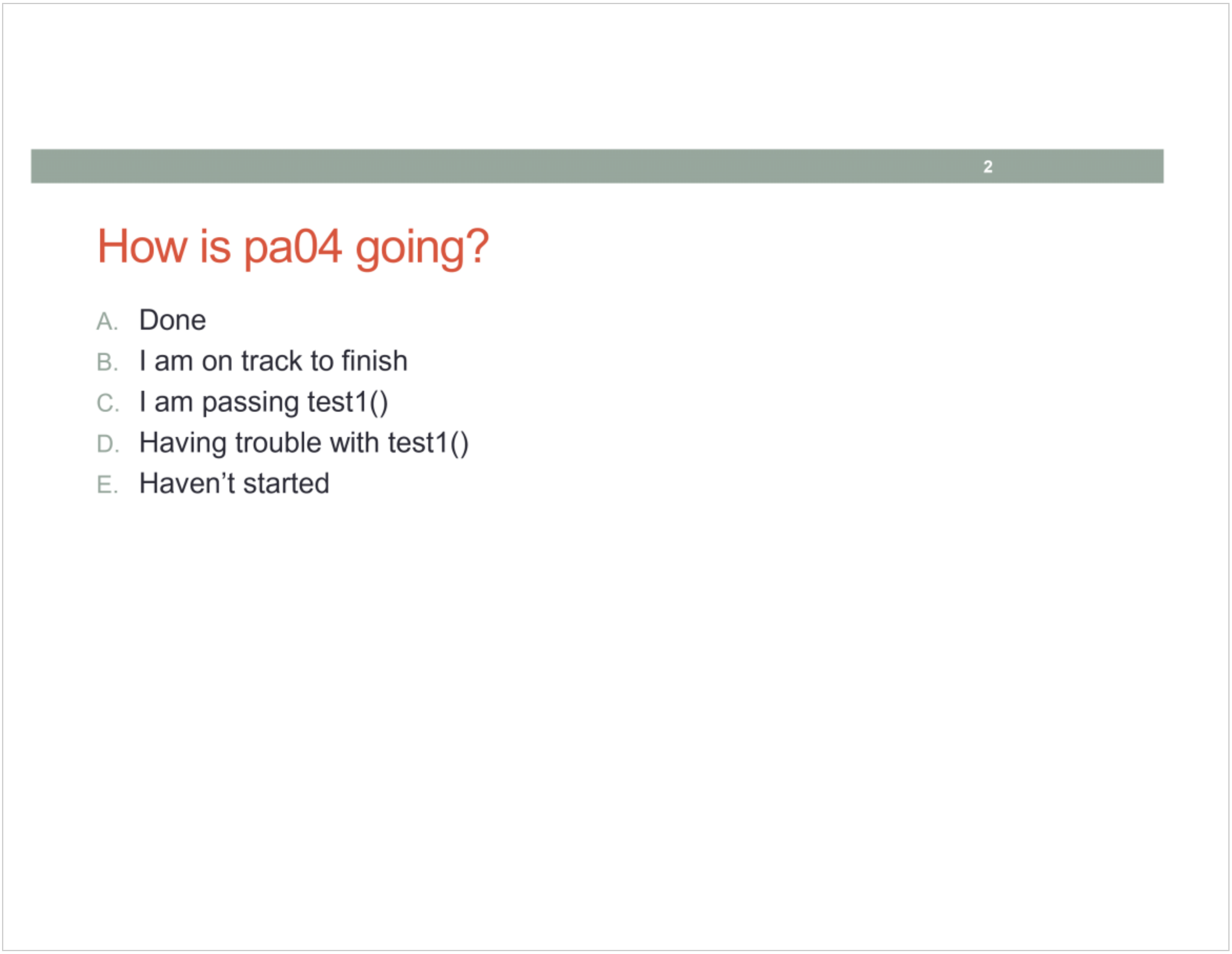
```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```

GitHub

Page 2

Monday, May 15, 2017 12:30 PM

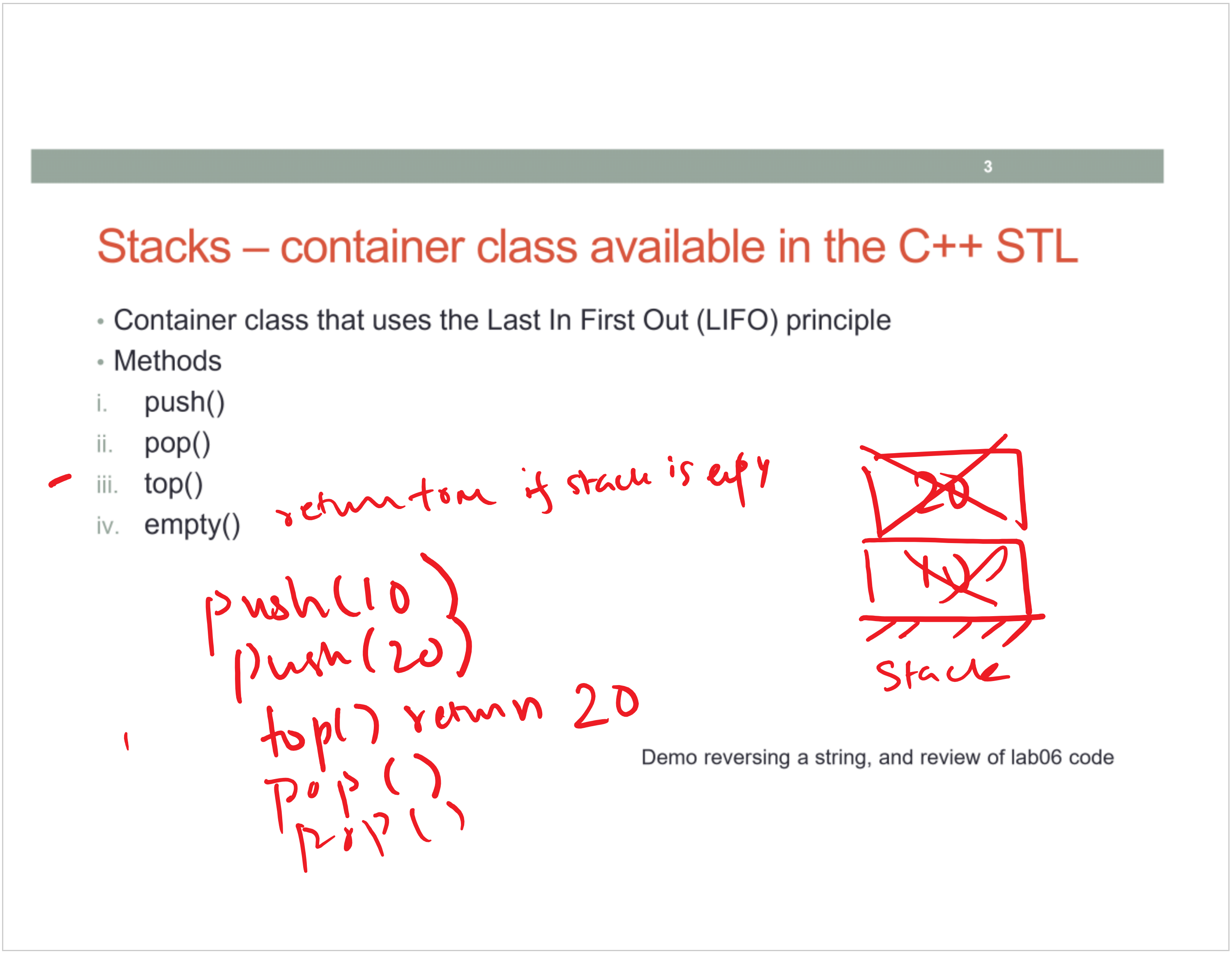


How is pa04 going?

A. Done
B. I am on track to finish
C. I am passing test1()
D. Having trouble with test1()
E. Haven't started

Page 3

Monday, May 15, 2017 12:30 PM



Stacks – container class available in the C++ STL

- Container class that uses the Last In First Out (LIFO) principle
- Methods

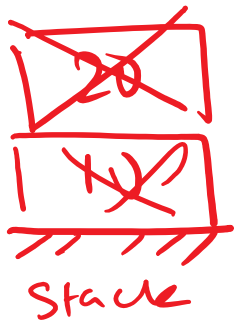
i. push()

ii. pop()

iii. top() return tom if stack is epty

iv. empty()

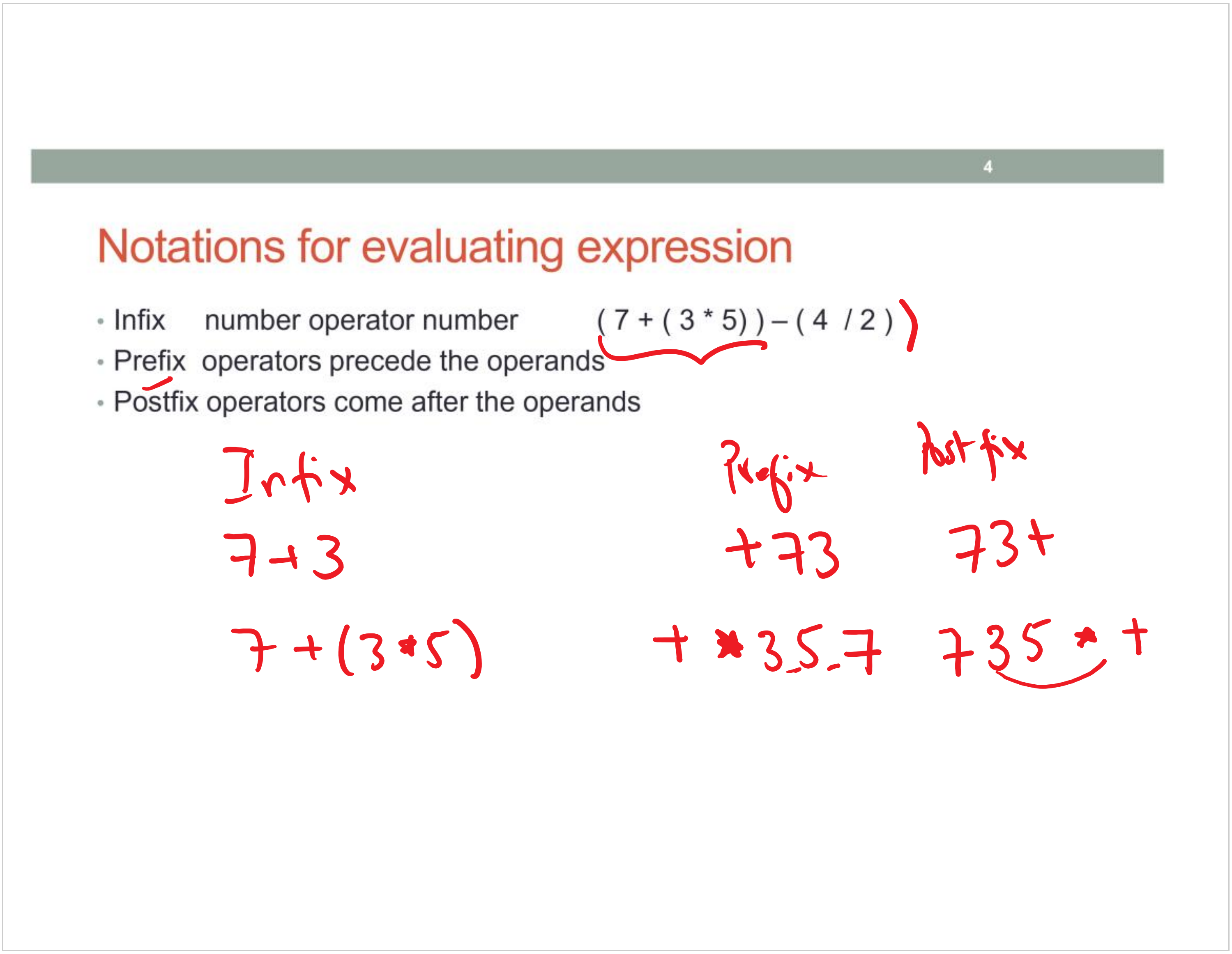
push(10)
push(20)
top() return 20
pop()
pop()



Demo reversing a string, and review of lab06 code

Page 4

Monday, May 15, 2017 12:30 PM



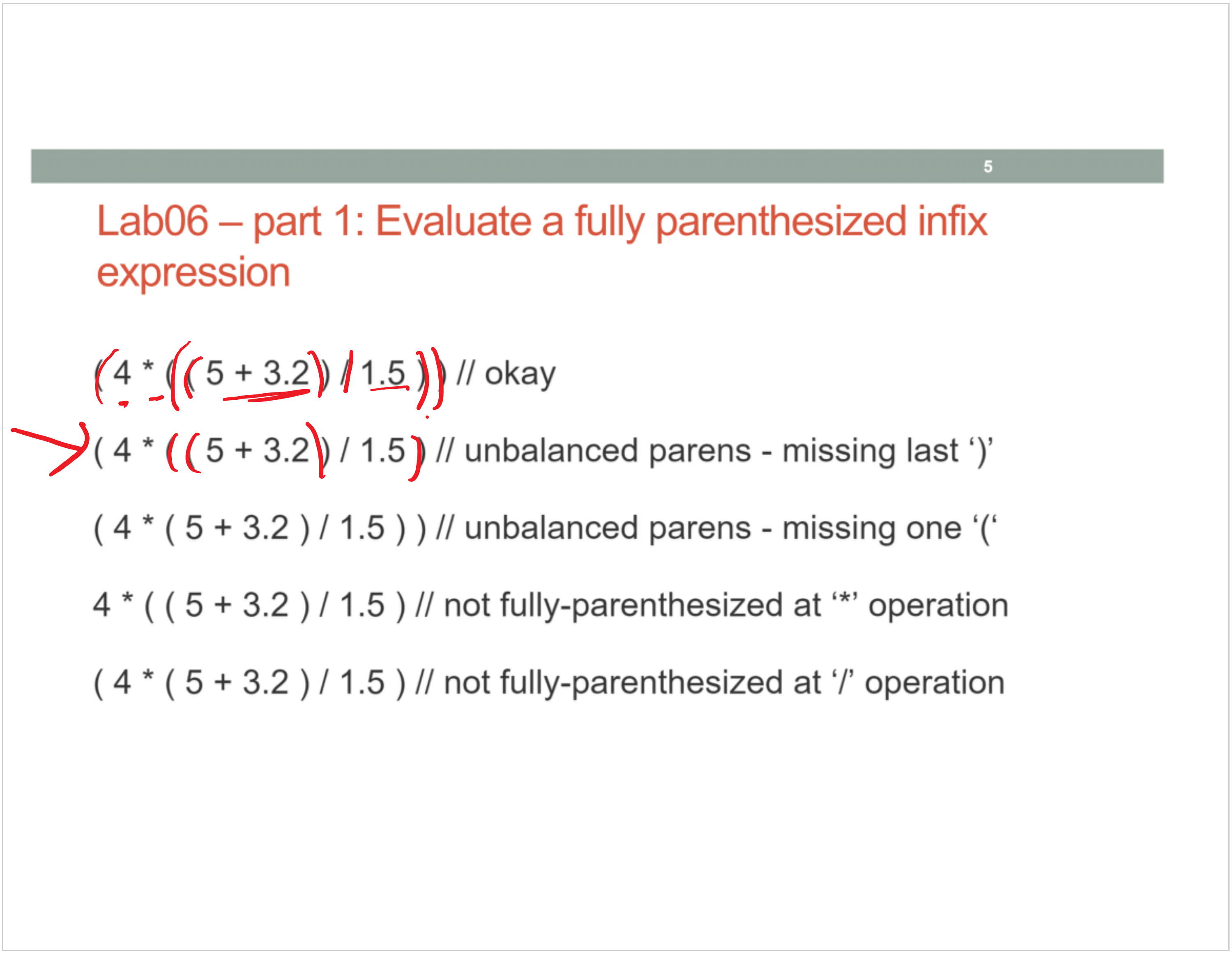
Notations for evaluating expression

- Infix number operator number $\underbrace{(7 + (3 * 5))}$ – (4 / 2))
- Prefix operators precede the operands
- Postfix operators come after the operands

Infix	Prefix	Postfix
7+3	+73	73+
7+(3*5)	+ * 3 5 7	7 $\underbrace{3 5 *}$ +

Page 5

Monday, May 15, 2017 12:30 PM



Lab06 – part 1: Evaluate a fully parenthesized infix expression

(4 * ((5 + 3.2) / 1.5)) // okay

(4 * ((5 + 3.2) / 1.5) // unbalanced parens - missing last ')'

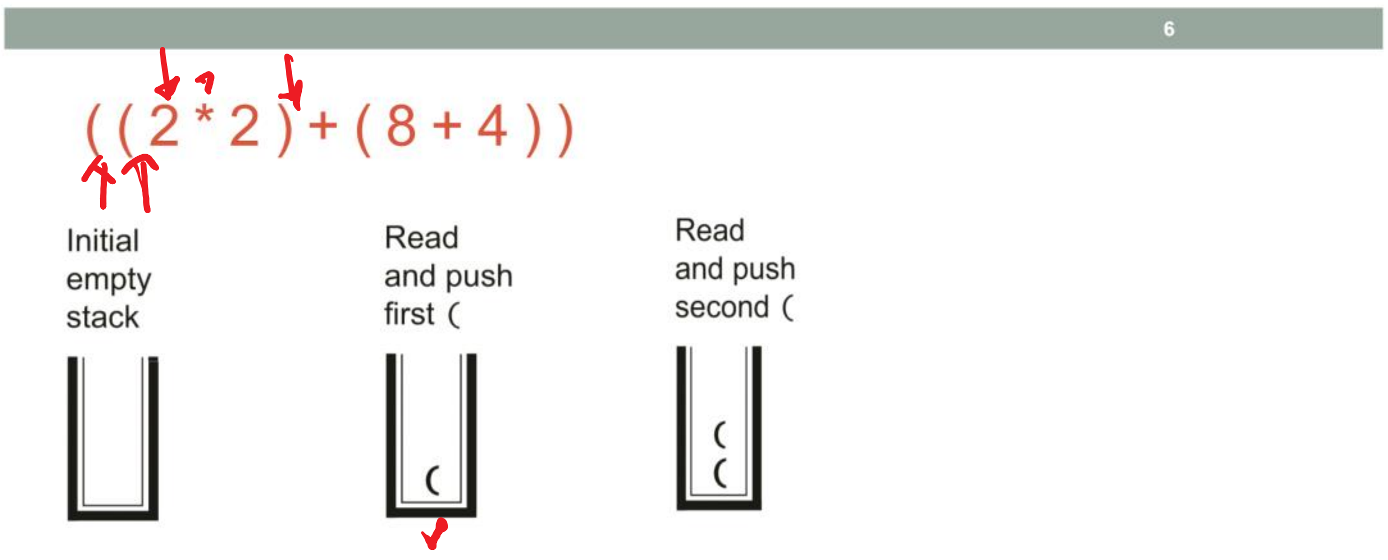
(4 * (5 + 3.2) / 1.5)) // unbalanced parens - missing one '('

4 * ((5 + 3.2) / 1.5) // not fully-parenthesized at '*' operation

(4 * (5 + 3.2) / 1.5) // not fully-parenthesized at '/' operation

Page 6

Monday, May 15, 2017 12:30 PM



((2 * 2) + (8 + 4))

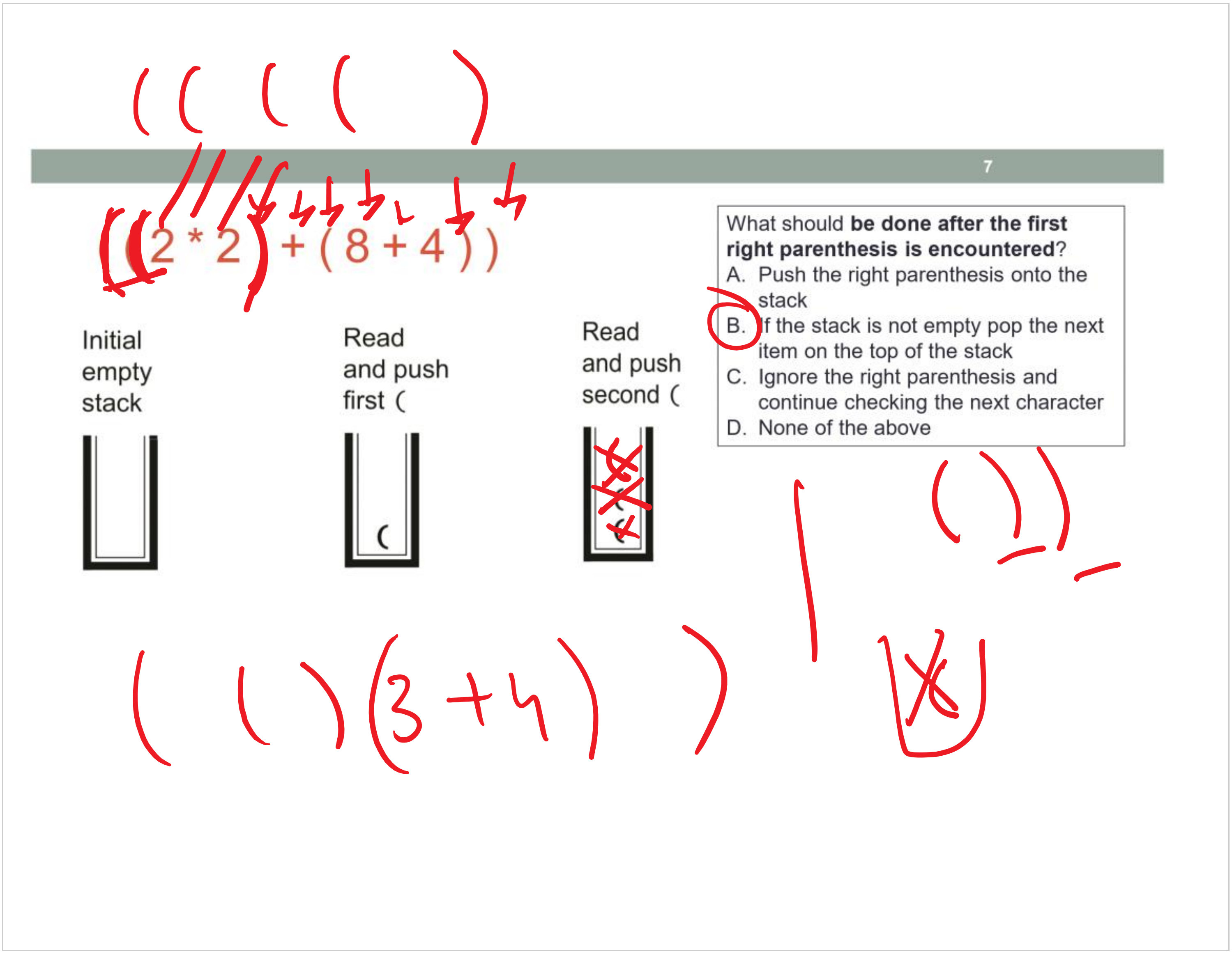
Initial empty stack

Read and push first (

Read and push second (

Page 7

Monday, May 15, 2017 12:30 PM



(((( )

((2 * 2) + (8 + 4))

What should **be done after the first right parenthesis is encountered**?

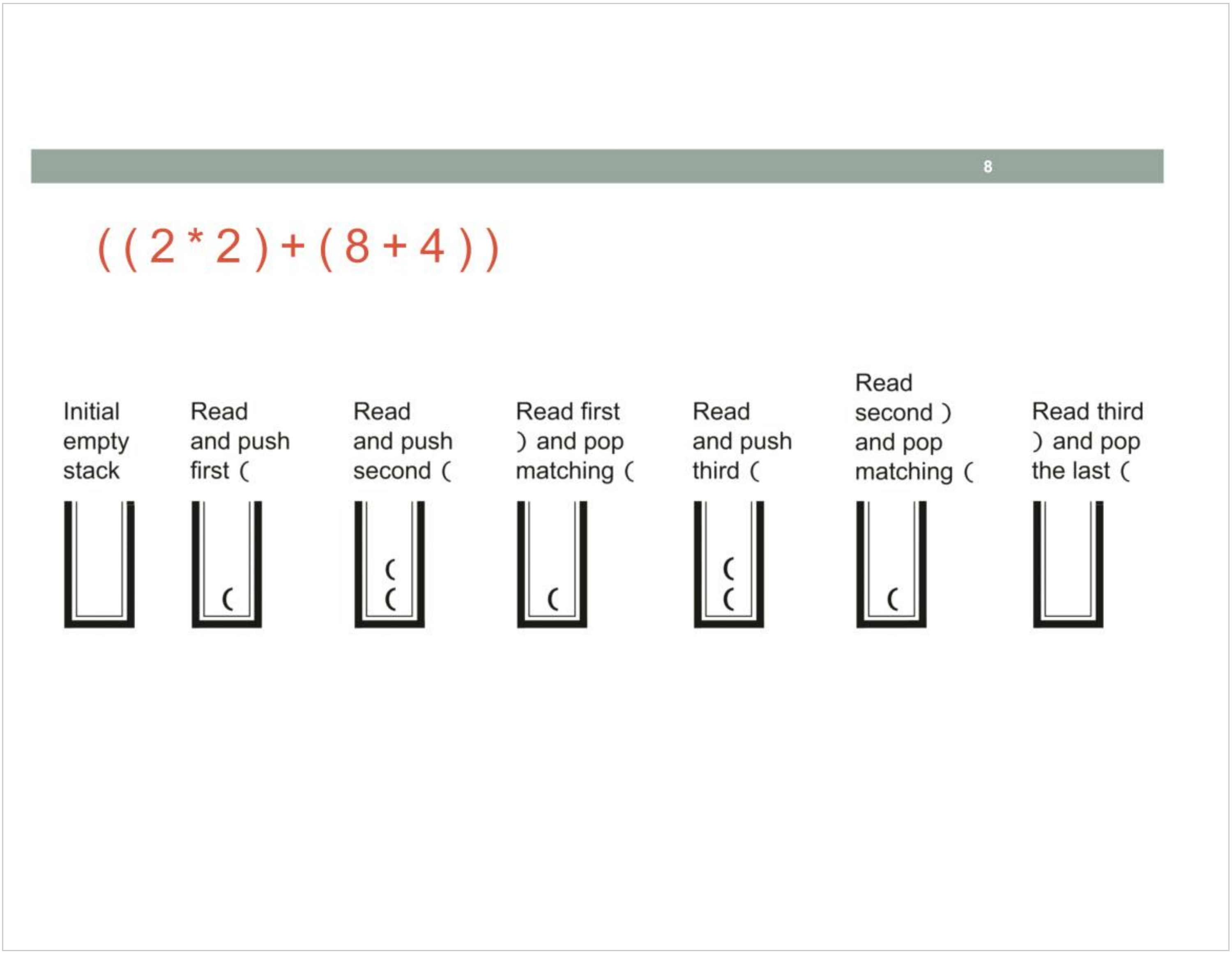
- A. Push the right parenthesis onto the stack
- [✓ B.] If the stack is not empty pop the next item on the top of the stack
- C. Ignore the right parenthesis and continue checking the next character
- D. None of the above

(( ))

(( ) (3 + 4))

Page 8

Monday, May 15, 2017 12:30 PM

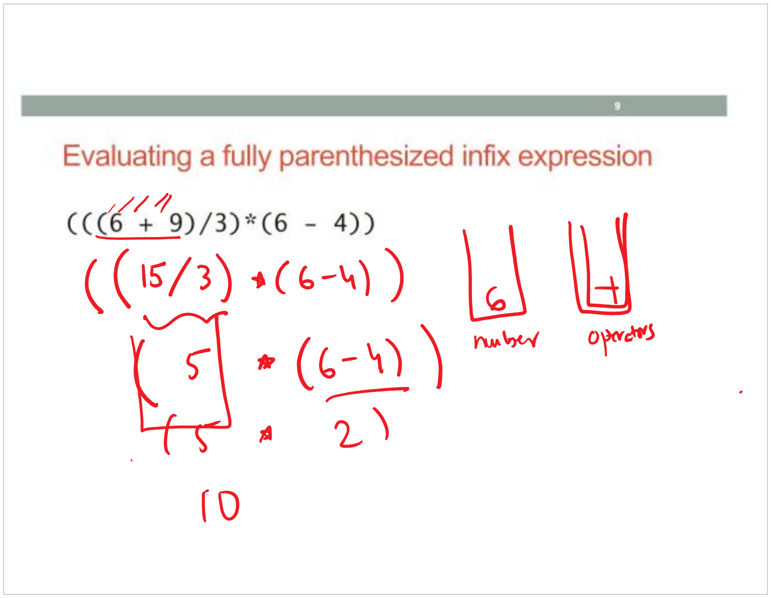


((2 * 2) + (8 + 4))

Initial empty stack
Read and push first (
(
Read and push second (
(
(
Read first) and pop matching (
(
Read and push third (
(
(
Read second) and pop matching (
(
Read third) and pop the last (

Page 9

Monday, May 15, 2017 12:30 PM



Evaluating a fully parenthesized infix expression

```
(((6 + 9)/3)*(6 - 4))
```

((15/3) * (6-4))

5 * (6-4)

(5 * 2)

10

6

number

+

operators

number

operator

6 + 9

((6 + (9/3)) * (6-4))

2

9

num

oper

9/3

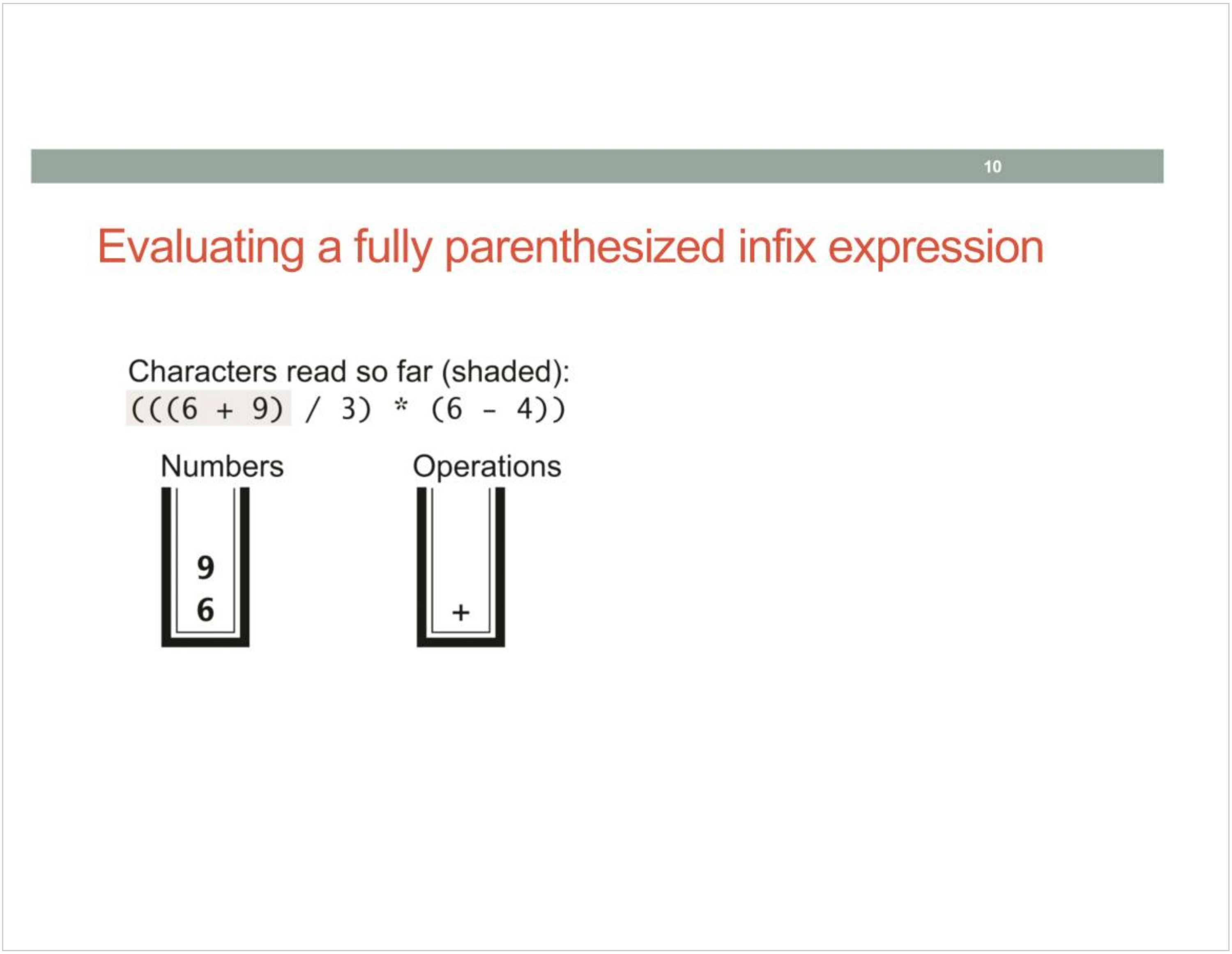
6+3

((12+6) / 7)

(= (_ 12 _ + _ 6 _)

Page 10

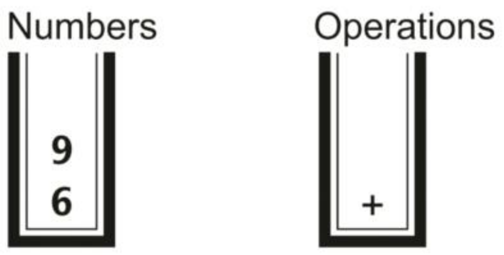
Monday, May 15, 2017 12:30 PM



Evaluating a fully parenthesized infix expression

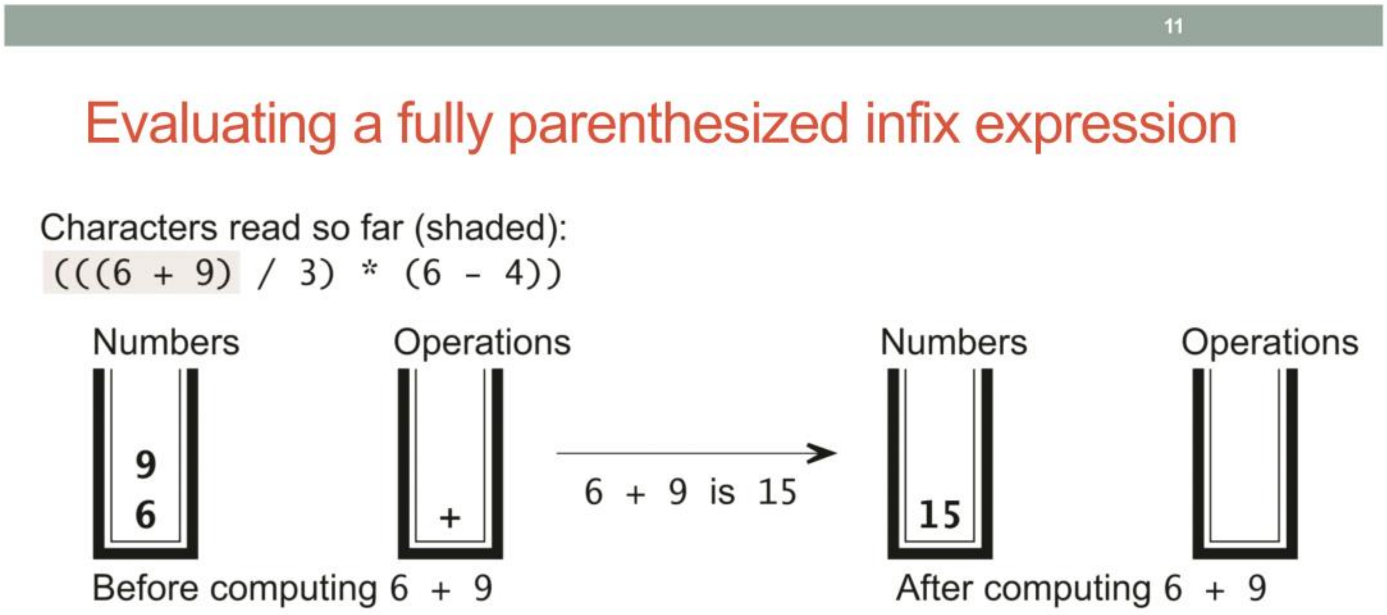
Characters read so far (shaded):

```
(((6 + 9) / 3) * (6 - 4))
```



Page 11

Monday, May 15, 2017 12:30 PM



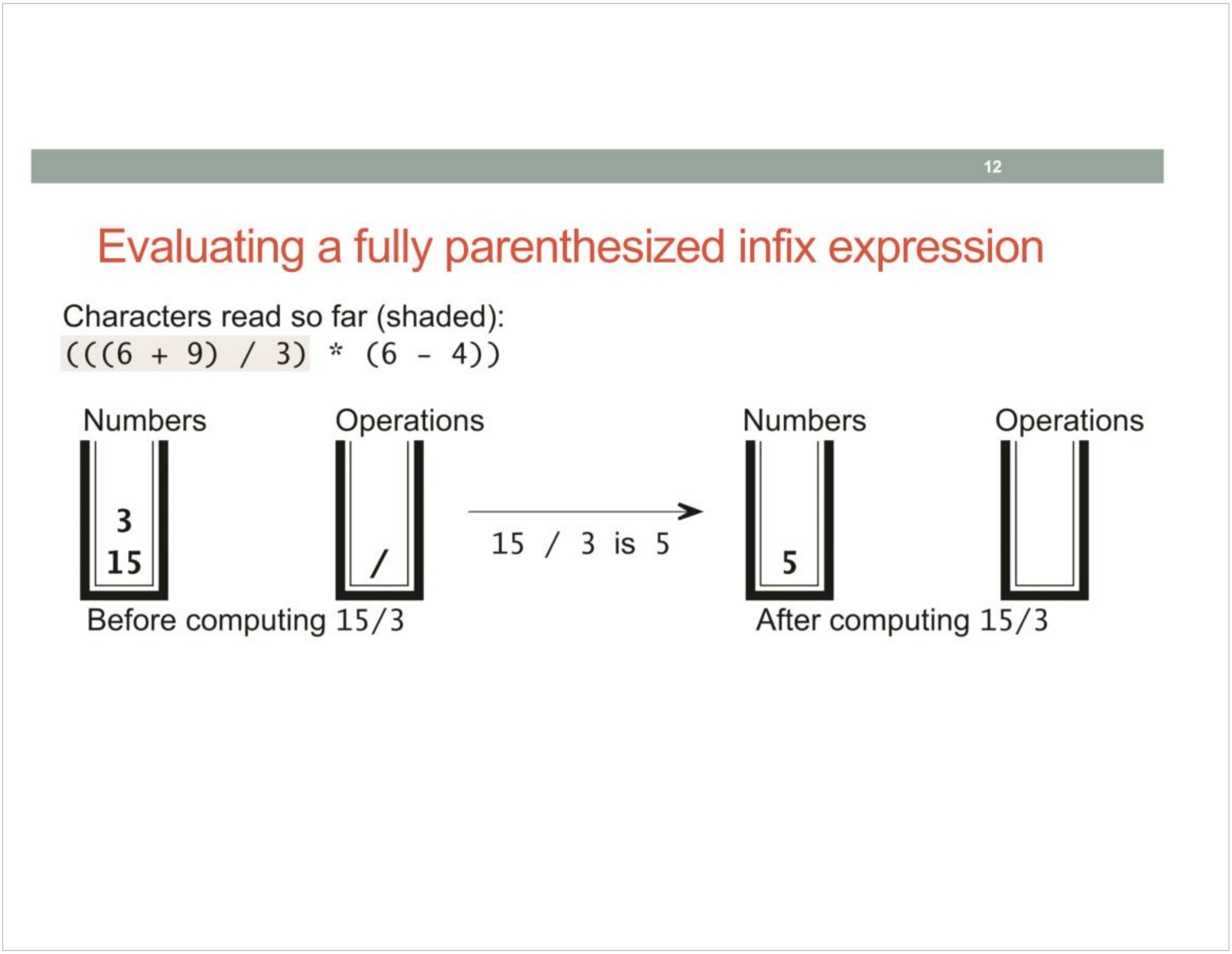
Evaluating a fully parenthesized infix expression

Characters read so far (shaded):

`(((6 + 9) / 3) * (6 - 4))`

Page 12

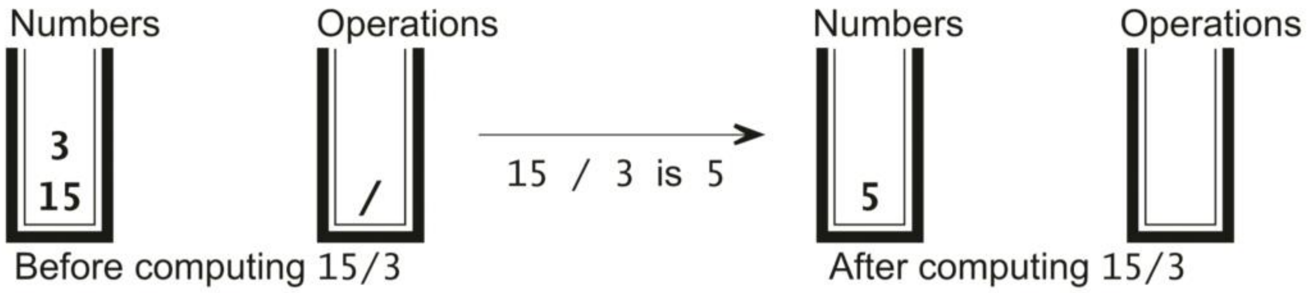
Monday, May 15, 2017 12:30 PM



Evaluating a fully parenthesized infix expression

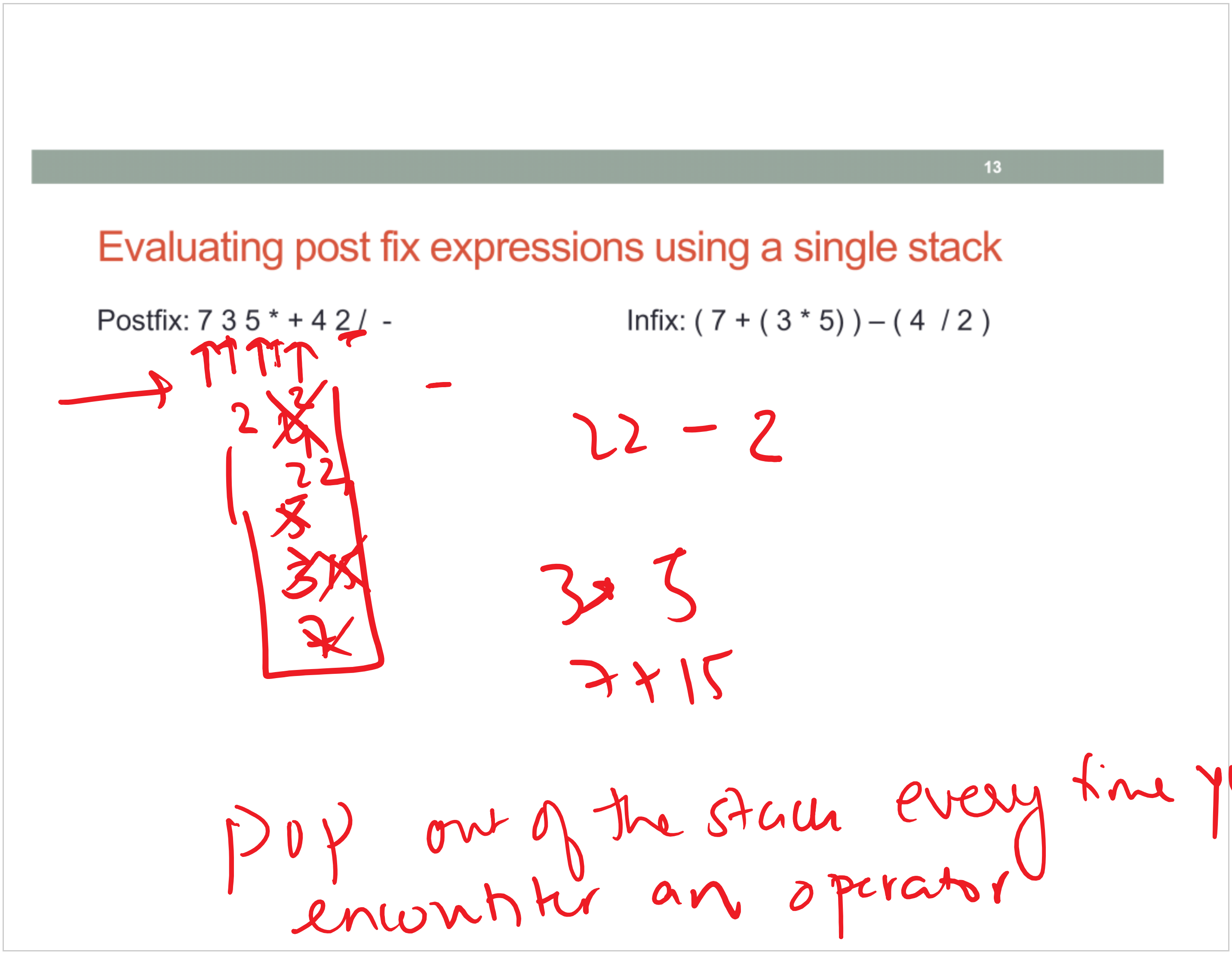
Characters read so far (shaded):

```
(((6 + 9) / 3) * (6 - 4))
```



Page 13

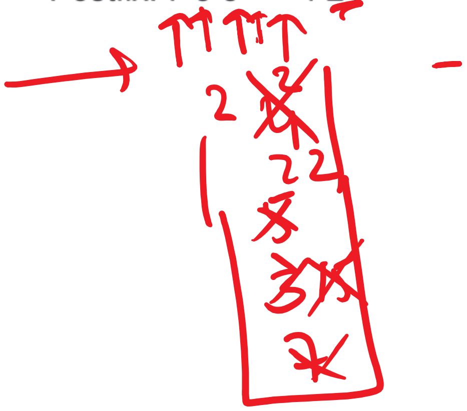
Monday, May 15, 2017 12:30 PM



Evaluating post fix expressions using a single stack

Postfix: 7 3 5 * + 4 2 / -

Infix: (7 + (3 * 5)) – (4 / 2)



22 - 2

3 * 5

7 + 15

Pop out of the stack every time you encounter an operator