Evaluating math

aka using Stacks to implement Dijkstra's "Shunting Yard Algorithm."

All the code is on Github if you want to follow along or try the challenges.

https://github.com/tylerprete/evaluate-math

What are we trying to accomplish?

- We'd like to accept mathematical expressions, such as "9 - (1 + 3) * 2" and evaluate them.
- In this case, that'd be 1.

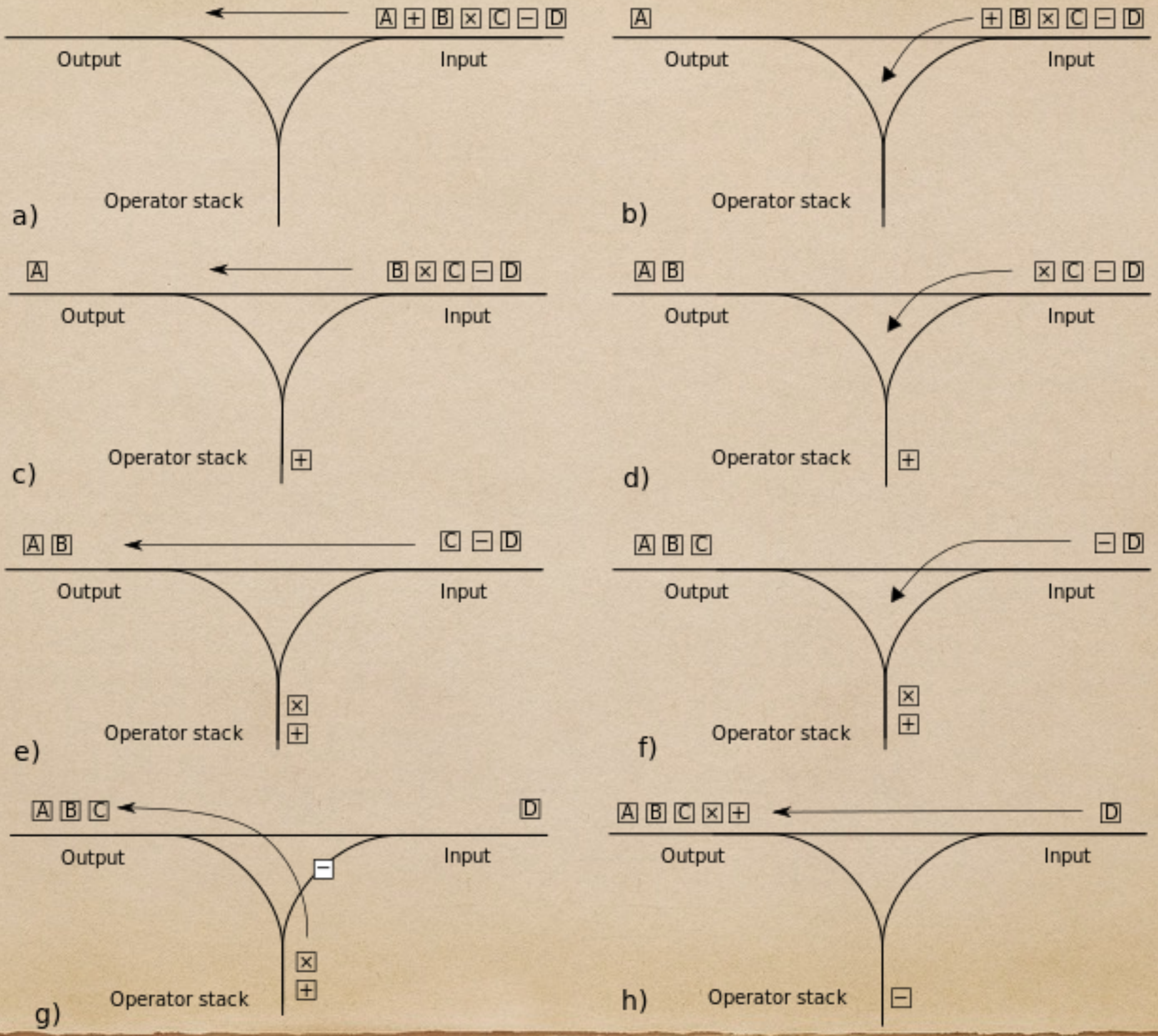
Luckily for us, in 1961, Dr. E.W. Dijkstra figured out how to do this.

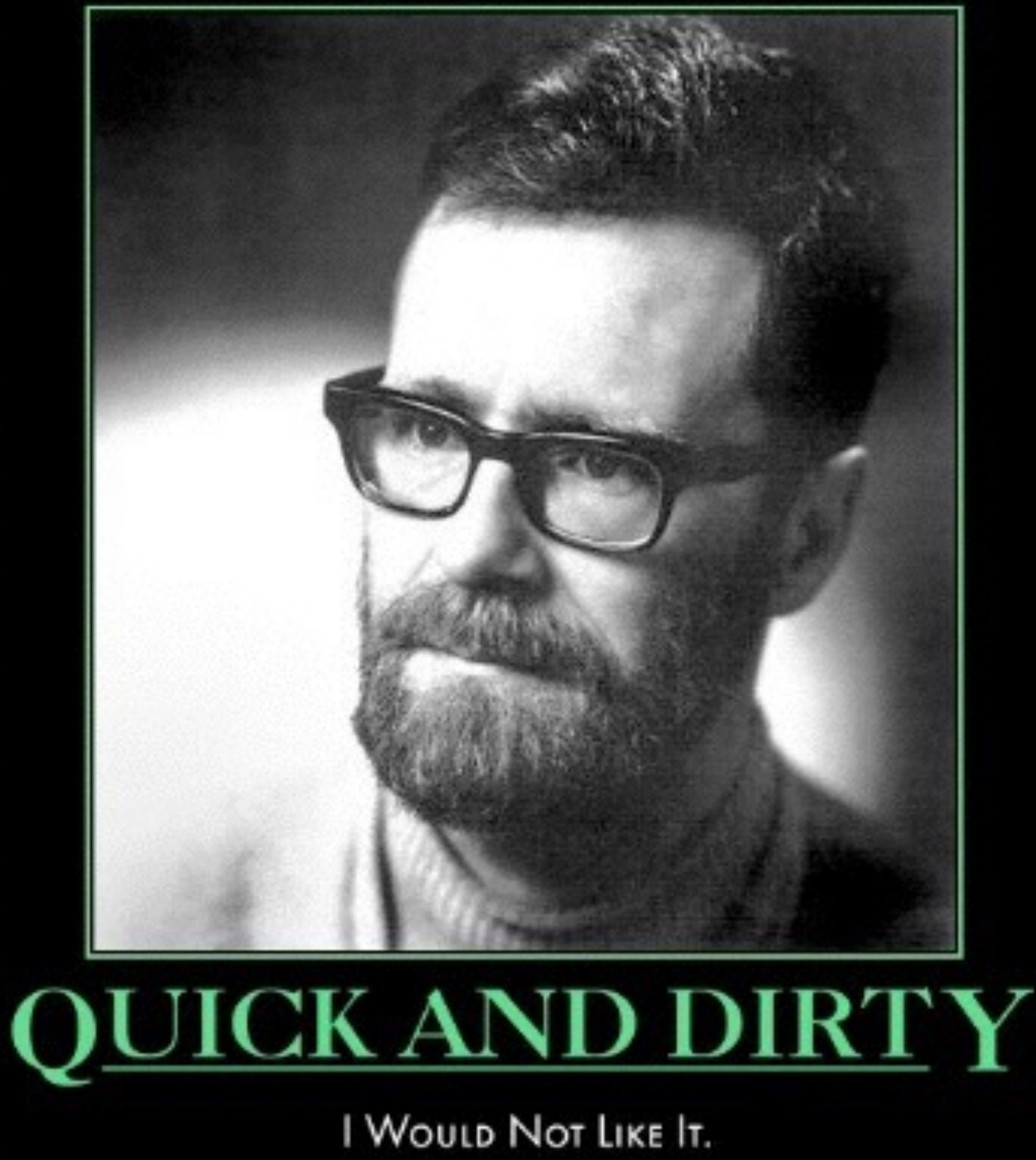
He'd called it the Shunting-Yard Algorithm, which is a British name for places where trains switch rails.

ECTRICALLY OPERATED SHUNTIN

n Hump Shunting Yard, on the Sou

Maybe you can see the resemblance?





Edsger W. Dijkstra

Object-oriented programming is an exceptionally bad idea which could only have originated in California

I don't know how many of you have ever met Dijkstra, but *you probably know that arrogance in computer science is measured in nano-Dijkstras. Alan Kay.*

But first, we'll take a detour

- Did anyone use an HP graphing calculator in school, rather than one of those TI-83s?
- They use a notation called Reverse Polish Notation (RPN), also known as Postfix.

Postfix (RPN)

- In normal (aka Infix) notation, we write “9 - (1 + 3) * 2”.
- In Postfix, we’d write that “9 1 3 + 2 * -”.

Why on earth would you do that?

- We get rid of precedence issues! Once it's written in postfix, we can evaluate rather simply.
- I'll give a demonstration on the whiteboard.

Introducing… The Stack!

Just like that stack of books, we can add items to a stack, but we can only get to the book on top.

Let's evaluate postfix!

- Demonstration time. For now we'll assume we can convert normal expressions to postfix. Let's use a stack to evaluate them!
- This'll be done live… I didn't feel like making slides for this.

Infix (the whole enchilada)

- Now that we can evaluate postfix, let's convert infix to postfix, and evaluate that.
- We solve a hard problem by first converting it to an easier one that we already know how to solve.
- This a common pattern in CS, and is something to watch out for.

Infix

- Once again, we'll be using a Stack, and we'll be writing this together. There's really just too much to go over in slides.

Challenge Problem!

- A similar problem to what we've gone over is detecting if parenthesis (and/or brackets) in an expression are balanced.
- "([ ])", for example, is balanced.
- "([ )]" and "(( )" are not.

Challenge Problem!

- I’ve written some tests for you if you’d like to give it a try. The files are in the balanced folder in the Github repo for this code.
- (https://github.com/tylerprete/evaluate-math)