**Combinational Logic I & II**

Brian Webb – ID: 34775506

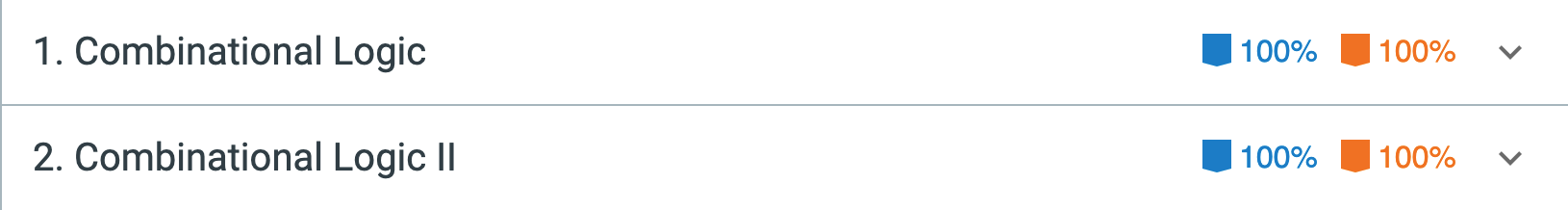
Student

CSC120-10187: Digital Design Fundamentals

Dinesh Sthapit / Carlos Rodriguez

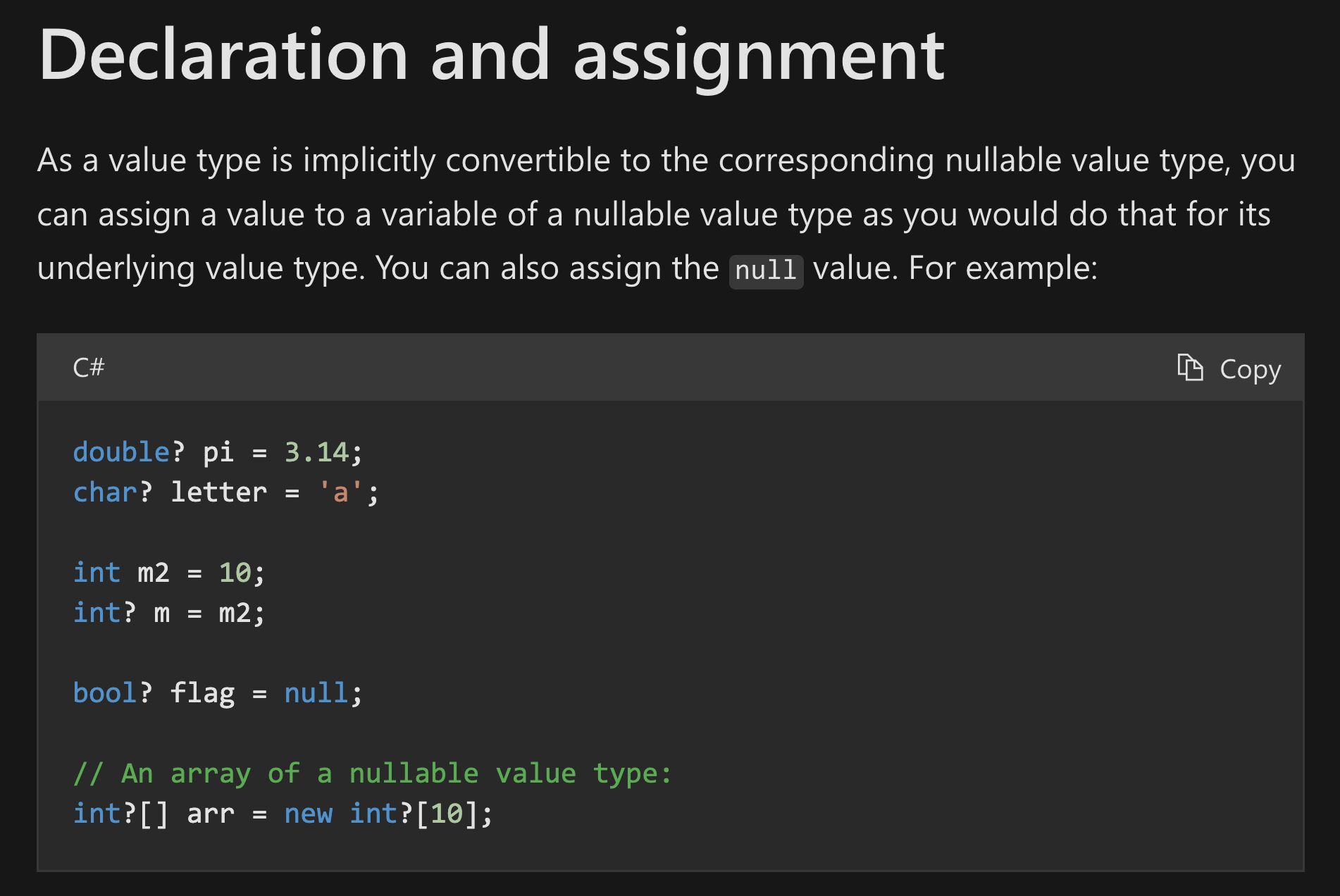
April 15, 2022

Note: Much of my module 1 included the C# from the original assignment. So the screenshots here reflect some of that. Here’s the screenshot of the two completed modules, with the original assignment screenshots below.



I needed to learn how to create a nullable type so I could have optional parameters for one of my functions:

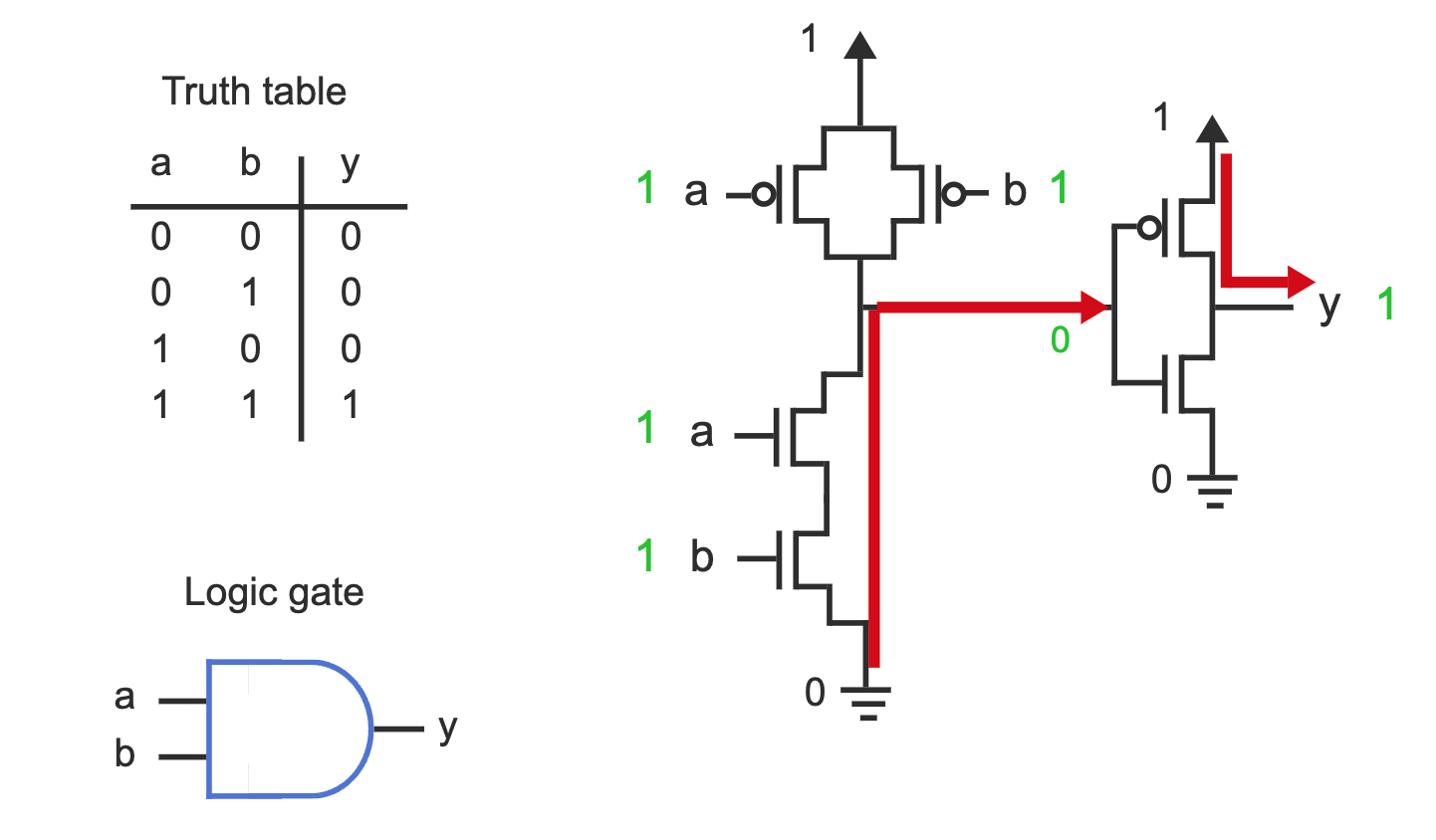
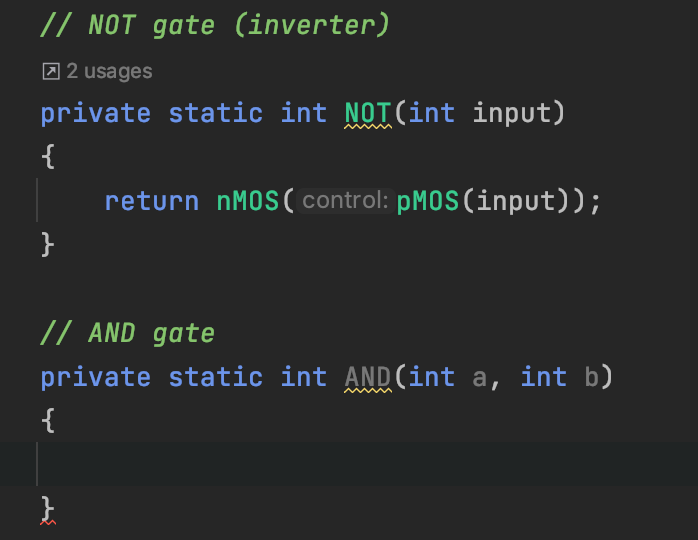
<https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/builtin-types/nullable-value-types>



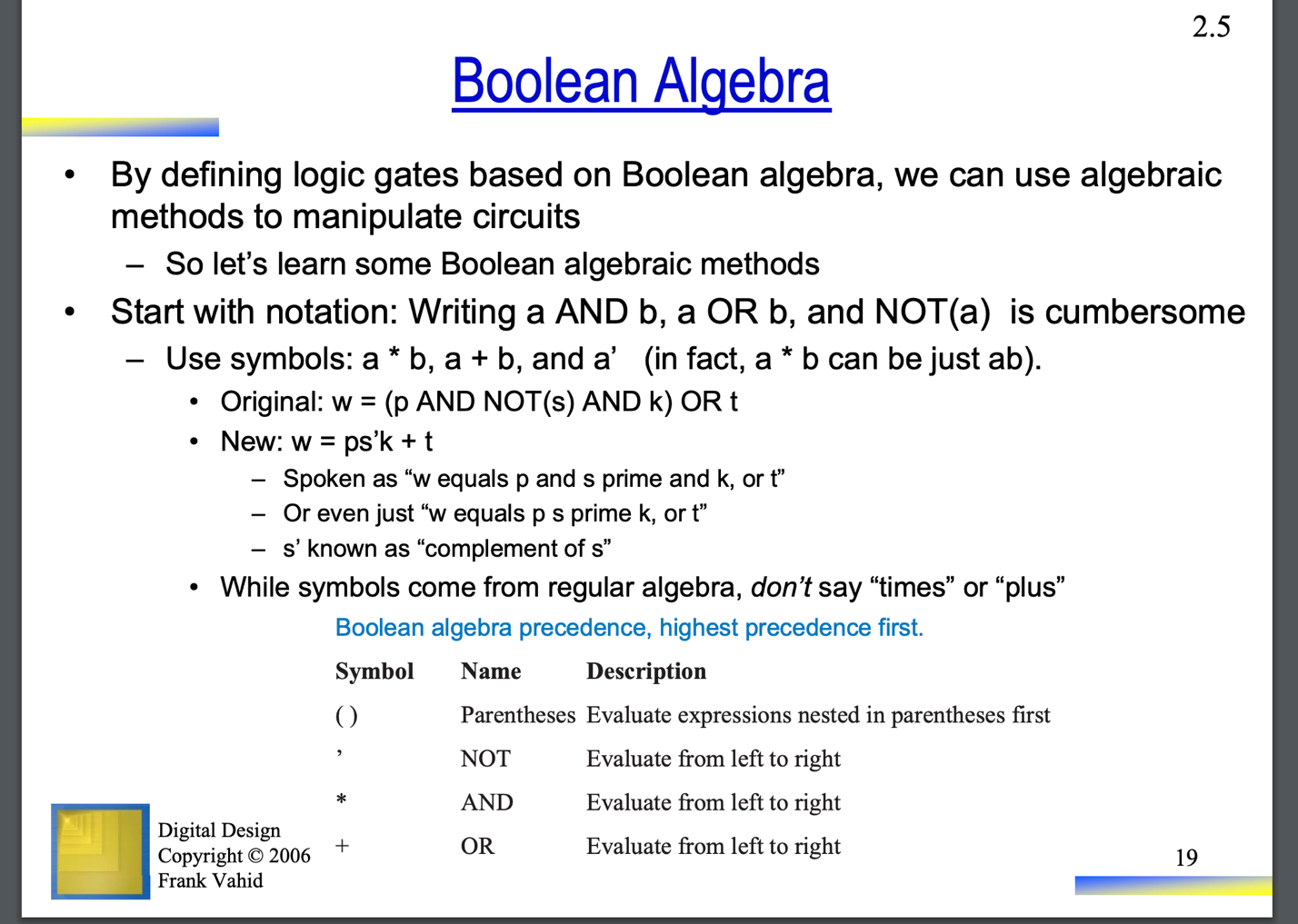
As I don’t use C# much, I also wanted to see how to throw an exception for these parameters; I found that here:

<https://docs.microsoft.com/en-us/dotnet/api/system.argumentexception?view=net-6.0>

I was attempting to create gate functions by using the nMOS and pMOS functios created previously… but I wasn’t understanding the “flow” or the order of how I would use these transitor functions to create the larger gates.



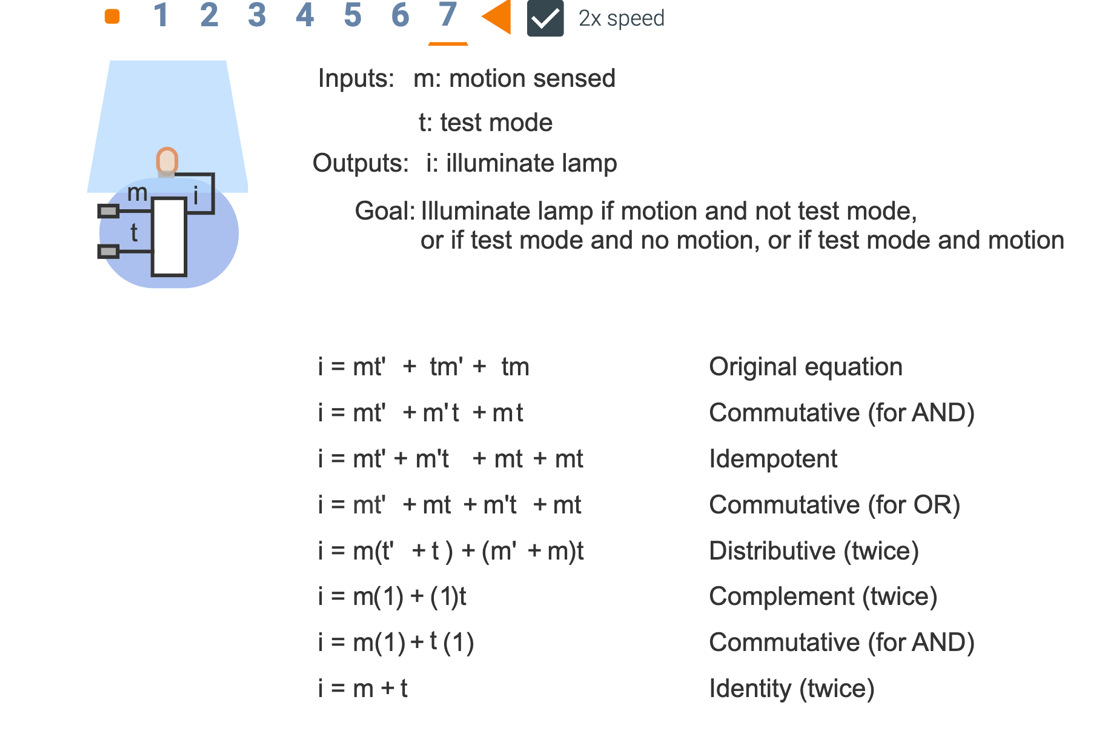
When the lesson mentioned the digital design shorthand, I was curious how that looked and found this… it didn’t quite match the symbols used in the lesson, but offered at least one example of how this could look.



I also found this link interesting, although I didn’t want to apply much in this fashion, I went a different route: <https://www.geeksforgeeks.org/program-to-implement-logic-gates/>

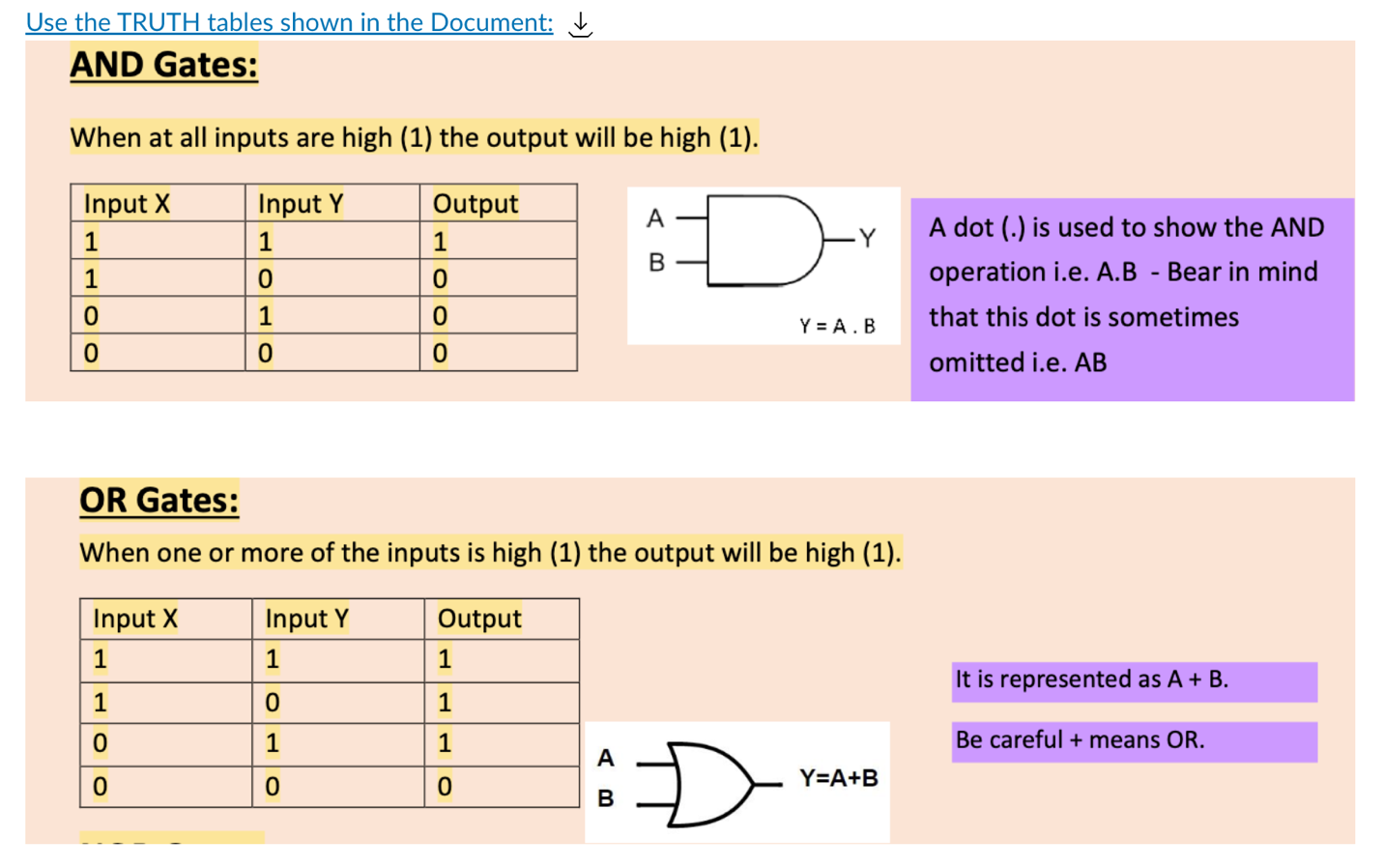
I ended up using only nMOS and pMOS transistors as well as other gates to create all the gates needed.

I found this super interesting.

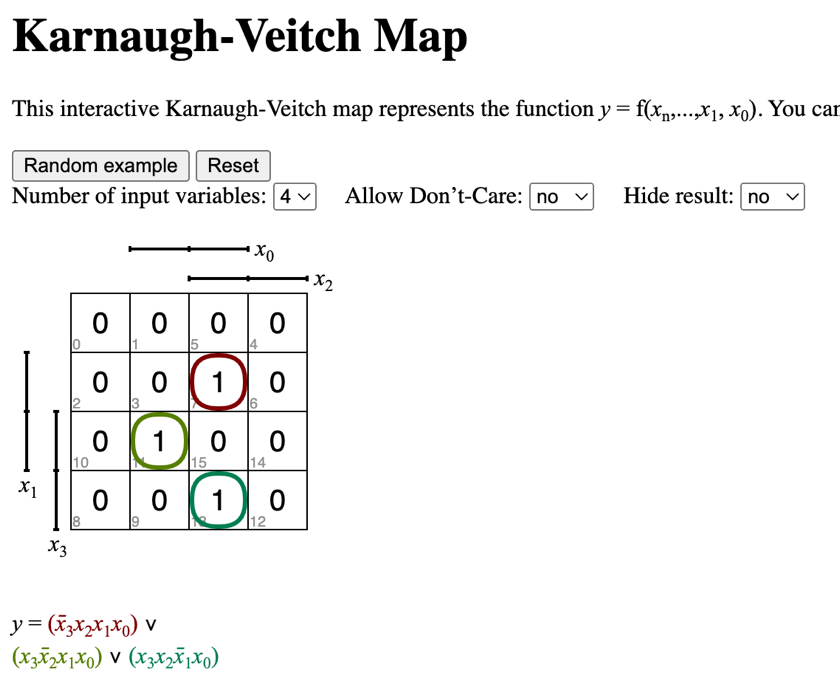


Brian Webb – ID: 34775506  
GW 2022 SPRING-CSC120 10187

I used this section of the lesson as a reference to confirm accuracy of my gate code.



I found this tool to be pretty cool!



This style of diagram seems like it would help significantly to implement this in code.

