Task 1

1) Uses comments, semi colons to end lines of code, key words like “and”, “not”, “case”, case sensitive, etc. It’s different because of the way sizing is done with choosing the size of the number, the number itself, the base format to be used, and then the actual number value
2) The Verilog data-type “reg” can be used to model hardware registers because it has the ability to hold values between assignments. Regs can be used to represent combinational logic.
3) Nested modules are not allowed. Because hardware is created in a hierarchical manner and Verilog aims to model that, creating nested modules does not work. They must be done one by one (top down, hierarchical).

Task 2

- **T2.1:** What was easy to explain? What was hard and why? - It was easy to explain that we are checking every possible input in this code. We check all the combinations of ones and zeros that can be inputted. It’s hard to understand now what is going on with the waves shown on the website. I think that a green wave signifies a one and a blue wave signifies a zero.
- **T2.2:** What aspects of Verilog seem more familiar to you? What makes other aspects new? - Words like “module” and “import” along with syntax like loops are like programming languages I have used in the past are familiar. The fact that we are working with hardware as a simulation is very new. It’s hard to explain why but it feels a lot less comfortable doing that than with software
- **T2.3:** Is this hardware development workflow convenient or inconvenient? List advantages and disadvantages. Describe possible alternatives. - It’s convenient because we are able to model complex hardware and make it abstract. It is inconvenient because learning Verilog seems to have a steep learning curve for people unfamiliar with hardware. There is not much of an alternative to this, it is just a difficulty you must struggle through.
- **T2.4:** What are the outputs like for each compilation step? Did you observe signals in GTKWave? If not, why do you think that is the case? - The outputs are green waves when a variable is one and blue waves when variables are zero. The lines start as blue at first when every variable is a zero. Then “e” becomes a one and turns into a green wave.

- **T2.5:** How do structural, data flow and behavioral code examples compare? When would you use one versus the others? - Data flow explains how data flows throughout a system. The focus tends to be on inputs and outputs. Behavioral code is used to describe behavior from digital circuits without specifying how circuits themselves are working. Structural code is used to specify how each piece of your circuit is being constructed. They are all used to solve different problems in different situations.