1. Write a Verilog code module to implement one-bit full adder. Write another module to implement eight-bit ripple carry adder that instantiates eight one-bit full adders and connects them properly. Now, add a test bench to test the eight-bit ripple carry adder.
Write a test bench to test the eight-bit adder. Make sure to display your inputs, sum, and carry out. Your test bench must have fifteen different inputs. Put five-time unit delay between consecutive inputs. Place one module in one Verilog file i.e., you will have three Verilog files.

2. Write a Verilog code to implement an eight-bit comparator using eight one-bit comparators. Follow the similar nomenclature as discussed in question 1. Your test bench must have fifteen different inputs. Put five-time unit delay between consecutive inputs. Place one module in one Verilog file i.e., you will have three Verilog files.

3. Write a Verilog code to implement a two-to-four decoder. Then a top module for three-to-eight decoder using two-to-four decoder. Write a test bench to test three-to eight decoder for all possible input with five-time delay.

4. write a Verilog code to implement an 8-to-3 priority encoder where priority is given to the least significant input position with a 1. For example, if the input is 11000000, the output would be 110 encoding the position of the least significant 1 in the input (leftmost bit is the most significant bit). Write a test bench to test encoder for at least five test cases. Put five-time unit delay between consecutive inputs.

Instruction:
Submit your code and other files inside in a zip file named clearly as described below.
G<Group No.>R_<Roll no.> A<Assignment No.>.zip
Example. G1R_200410_200411_200412.zip

Any information/assumption about implementation should be described clearly in a separate text file.
Submit your iverilog/verilog files named clearly as described below.
A<Assignment No.>Q<Question No.>_<top module name>.v
Example. A1Q1_one_bit_full_adder.v