1. Recall the construction that we saw in the lecture when we discussed NP-hardness of verifying DNNs with ReLU nodes. We saw different gadgets for disjunction, negation, and conjunction, that can allow us to encode 3-SAT instances as DNN verification problems. We made the assumption that the input to the disjunction and negation gadgets should be either 0 or 1.

   (a) Complete this construction so that it works when the inputs are in the range $[0, 1]$, and explain why your construction is correct. [*Hint*: There is a discussion on this in the Reluplex paper (in the appendix), but it is not complete. You can get the idea from there but you will need to work out the details.]

   (b) Recall that a DNN verification query is a 3-tuple, consisting of the *precondition*, the *network*, and the *postcondition*. Construct the DNN verification query corresponding to the following 3-SAT instance.

   $$(x_2 \vee \neg x_3 \vee x_1) \wedge (\neg x_3 \vee \neg x_2 \vee x_5) \wedge (\neg x_5 \vee x_3 \vee x_2) \wedge (\neg x_4 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee \neg x_4 \vee \neg x_2)$$

   (c) Encode this verification query in the input language of Z3 and submit the corresponding `.smt2` file along with your assignment.

   (d) Use Z3 to get a solution to the DNN verification query, and explain how you can obtain a satisfying assignment for the above 3-SAT instance from that.

2. Consider the toy network shown in Fig. 2 in the Reluplex paper.

   (a) Find out, using the Reluplex algorithm, whether it is possible for the output neuron to get a value in the range $[2, 3]$ if the input neuron must be in the range $[1, 2]$.

   (b) Perform the same task as above, with one algorithmic change: when Simplex returns and a ReLU is found broken, split on (one of) the broken ReLU, and solve the problem instances thus obtained. Between the two branches, solve that branch first where the forward facing variable is assumed to be 0 (i.e., the corresponding backward facing variable has been assumed to be ≤ 0).