Due: Friday, October 29, 8AM EST

This assignment is worth 15% of final grade. Each question is worth 10 points. If you have no idea how to answer a question (or part of a question), you will receive 20% of the credit for that question (or subquestion) leaving the question (or subquestion) blank. If your answer makes no sense, you will not receive any credit. Any answer that shows some understanding of the question will receive some credit.

1. The following definitions apply to any *tree* but we will restrict attention to search trees as in the data structure we described for the dictionary data type. (These definitions will also be discussed in the Friday, October 15 tutorial.)

   - A search tree is a *binary search tree* if every node has at most two children.
   - A search tree for a set of n distinct items is *strictly balanced* if for some ℓ every leaf is at depth at most ℓ and there is no other search tree for (n items) for which the depth of every leaf is less than ℓ. (Note: This is perhaps not a standard definition. Note also that I am picturing the *root* of the search tree to be at the top and the leaves at the bottom. If you picture a search tree with the root at the bottom then you would discuss the height of a node.)

   (a) (5 pts) Draw a strictly balanced binary search tree for the following IDs: $1, 12, 15, 19, 23, 28, 40$. What is the depth ℓ of the leaves?

   (b) (5 pts) How would you find the smallest ID in binary search tree?

   (c) (5 pts) Suppose another three IDs are added to the balanced search tree iin part (a) and the tree is re-rebalanced, what will be the maximum depth of any leaf?

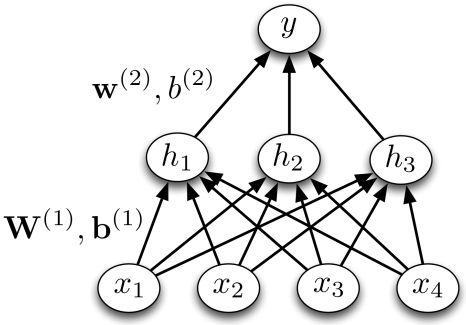
   (d) (5 pts) Can you characterize for what values of n will all leaves have the same depth in a balanced binary search tree?

2. (20 pts) Note: We will be explaining the notation for this question in the October 15 tutorial and (if necessary) the Monday October 18 class.

   In this question, you need to find a set of weights and biases for a neural net (with one hidden layer as below) for computing the following function f:
   the input consists of 4 (say rational) inputs $x_1, x_2, x_3, x_4 : x_1 \leq x_2 \leq x_3 \leq x_4$; the output $y = f(x_1, x_2, x_3, x_4)$ is

$$fx_1, x_2, x_3, x_4) = \begin{cases} 1 & \text{if } x_i \neq x_j \text{ for } i \neq j \\ 0 & \text{otherwise} \end{cases}$$

   You will use the following architecture.



   All of the hidden units and the output unit use a hard threshold activation function:

$$\phi(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

   Provide a set of weights and biases for h_1, h_2, h_3 and y so that the network implements the function f.

3. (20 pts) This question introduces an imnportant concept, namely when one problem can be *reduced* to another problem. (For those familiar with subroutines, this should already be a meaningful concept.

   Recall that by algorithmically solving a problem, we mean that we have a Turing machine program that halts on every input and outputs the correct answer. For $\{$YES, NO$\}$ decision problems we say we can say a problem is *decidable* or *undecidable* depemnding on whether or not the problem can be algorithmically solved.

   If A and B are two decision problems, we will informally define $A \leq_T B$ to mean "If problem B is decidable, then problem A is decidable. The equivalent contrapositive of this statment is "if problem A is undecidable then problem B is undecidable".

   In class we have stated that given an arbitrary program P and an input x to problem P, that it is undecidable to determine if P halts on input x. Using this version of the halting problem as a fact, show that given an arbitrary program Q, it is undecidable to determine if Q halts on all inputs.