*Print Your Name*:__________________________________________________________

Please sign the following pledge:  I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination.

Signed:______________________________________________________________

==================DO NOT WRITE BELOW THIS LINE====================
Part I (45 pts) Consider the neural network shown below, with activation function ReLU, and cost/error function C being the squared difference:

![Neural Network Diagram]

a. Write the cost function C as an expression in terms of \( a_1^2 \) and desired (target) output \( y \):

\[
C(a_1^2, y) = (a_1^2 - y)^2
\]

b. Find the partial derivative of C with respect to \( a_1^2 \):

\[
\frac{\partial C}{\partial a_1^2} = 2(a_1^2 - y)
\]

c. Using the function-notation “ReLU”, write \( a_1^2 \) as a function of \( z_1^2 \):

\[
a_1^2 = \text{ReLU}(z_1^2)
\]

d. Write \( z_1^2 \) as a function of \( a_0^1, a_1^1, w_{01}^1, w_{11}^1 \):

\[
z_1^2 = a_0^1 w_{01}^1 + a_1^1 w_{11}^1
\]

e. Express your result in part d above as an inner (dot) product of two vectors:

\[
a_1^1 \cdot w_1^1 \quad \text{or} \quad (a_0^1, a_1^1) \cdot (w_{01}^1, w_{11}^1)
\]

f. Using \( \frac{\partial C}{\partial w_{ij}^l} \) and learning rate \( \alpha \), write the formula for the new value of an arbitrary initial weight \( w_{ij}^l \) that one application of gradient descent would produce:

\[
w_{ij}^l = \alpha \frac{\partial C}{\partial w_{ij}^l}
\]

In parts g-i below, assume initial weights all equal to 1, and use a training pair \((x, y) = ((1, 1), 10)\), where \( x = (1, 1) \) is the input vector, and \( y = 10 \) is the desired (target) output value.

g. The numerical total weighted input \( z_1^1 \) to unit 1 in layer 1 will be: 3

h. The numerical total weighted input \( z_1^2 \) to unit 1 in layer 2 will be: 4

i. The numerical cost/error C for the given training pair is: \((4 - 10)^2 = 36\)
Part II (15 pts) Prove the 4th backprop equation (shown below); show all steps and give an explanation for each and every step (e.g., state a definition, rule, principle, reasoning, etc.):

\[
\frac{\partial C}{\partial w_{jk}^l} = a_j^l \delta_{k}^{l+1}
\]

(copied directly from the solutions to Hwk 3)

Working on the LHS, we have by the chain rule:

\[
\frac{\partial C}{\partial w_{jk}^l} = \sum_n \frac{\partial C}{\partial z_{n}^{l+1}} \left( \frac{\partial z_{n}^{l+1}}{\partial w_{jk}^l} \right)
\]

\[
= \sum_n \delta_{n}^{l+1} \partial \left[ \sum_m a_m^l w_{mn}^l \right] / \partial w_{jk}^l
\]

\[
= \delta_{k}^{l+1} \frac{\partial a_j^l w_{jk}^l}{\partial w_{jk}^l}
\]

(since the inner sum has zero derivative wrt \( w_{jk}^l \) except when m=j and n=k; this also uses the fact that activations \( a^l \) depend on the w's in layer l-1, not those in layer l)

\[
= \delta_{k}^{l+1} a_j^l
\]

which is equal to the RHS.
Part III (10 pts) Explain why bias units are important; i.e., what can go wrong without them? (35 words MAX!)

Without bias units, certain input-output relations might not be learnable. For instance, if the activation function is ReLU, such a nnet cannot output a non-zero result if the input data are all zeros.
Part IV (10 pts)  In some forms of *unsupervised* ("deep") nnet learning, (i) certain hidden layers have significantly fewer units than do the input and output layers, (ii) the output layer activations are compared to the input, and (iii) weights are adjusted to minimize this compared difference over many different input patterns. Give an intuitive explanation (35 words MAX!) of what (and how) this learns about inputs.

Less units in a hidden layer means input data might not be fully reproducible at the output layer. When output optimally matches input then high-level input features were computed in hidden layers.

Part V (20 pts)  Use PL refutation-resolution to prove the tautology $P \vee \neg P$. **Explain what you are doing in every step!**

Negate desired result: $\neg(P \vee \neg P)$, which has CNF equivalent $\neg P \wedge P$.

This then breaks into two clauses:

\[
\begin{align*}
\neg P \\
P
\end{align*}
\]

which resolve to give nil (the box).