This is a CLOSED-BOOK exam. You may use one sheet (front and back) of handwritten notes.

No calculators are permitted. You need not simplify explicit numerical expressions.

There are a total of 100 points in the exam. Each problem specifies its point total. Plan your work accordingly.

You must SHOW YOUR WORK to get full credit.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
</tr>
</tbody>
</table>

Name: ______________________________________________
Possibly Useful Formulas

Neural Nets

\[ a_k = u_k + \sum_j w_{kj} x_j \]
\[ y_k = g(a_k) \]
\[ \frac{\partial E}{\partial x_j} = \sum_k w_{kj} g'(a_k) \frac{\partial E}{\partial y_k} \]

Logistic Function

\[ \sigma(x) = \frac{1}{1 + e^{-x}}, \quad \sigma'(x) = \sigma(x) (1 - \sigma(x)) \]

Loss Functions

\[ E_{MSE} = \frac{1}{n} \sum_{i=1}^{n} \| \vec{z}_i - \vec{\zeta}_i \|^2 \]
\[ E_{CE} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{\ell=1}^{r} \zeta_{i\ell} \ln z_{i\ell} \]
\[ E_{CE} = -\frac{1}{n} \sum_{i=1}^{n} (\zeta_i \ln z_i + (1 - \zeta_i) \ln(1 - z_i)) \]

Affine Transform

\[
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix} =
\begin{bmatrix}
    a & b & c \\
    d & e & f \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    u \\
    v \\
    1
\end{bmatrix}
\]

Barycentric Coordinates

\[
\begin{bmatrix}
    x_0 \\
    y_0 \\
    1
\end{bmatrix} =
\begin{bmatrix}
    x_1 & x_2 & x_3 \\
    y_1 & y_2 & y_3 \\
    1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
    \lambda_1 \\
    \lambda_2 \\
    \lambda_3
\end{bmatrix}
\]

LSTM

\[ \tilde{i}[n] = \text{input gate} = \sigma(B_i \vec{x}[n] + A_i \vec{c}[n - 1]) \]
\[ \tilde{o}[n] = \text{output gate} = \sigma(B_o \vec{x}[n] + A_o \vec{c}[n - 1]) \]
\[ \tilde{f}[n] = \text{forget gate} = \sigma(B_f \vec{x}[n] + A_f \vec{c}[n - 1]) \]
\[ \vec{c}[n] = \tilde{f}[n] \odot \vec{c}[n - 1] + \tilde{i}[n] \odot g (B_c \vec{x}[n] + A_c \vec{c}[n - 1]) \]
\[ \vec{y}[n] = \tilde{o}[n] \odot \vec{c}[n] \]
Problem 1  (16 points)

In class, we have been working with the exponential softmax function, but other forms exist. For example, the polynomial softmax function transforms inputs $b_\ell$ into outputs $z_\ell$ according to

$$z_\ell = \frac{b_\ell^p}{\sum_k b_k^p},$$

for some constant integer power, $p$. The cross-entropy loss is

$$E = -\sum_\ell \zeta_\ell \ln z_\ell, \quad \zeta_\ell = \begin{cases} 1 & \ell = \ell^* \\ 0 & \text{otherwise} \end{cases}$$

Find $\frac{\partial E}{\partial b_j}$ for all $j$. 
Problem 2  (17 points)

Suppose you have a 10-pixel input image, \( x[n] \). This is processed by a one-pixel “convolution” (really just multiplication by a scalar coefficient, \( w \)), followed by a stride-2 max pooling layer, thus:

\[
\begin{align*}
ad[n] &= wx[n], \quad 1 \leq n \leq 10 \\
y[k] &= \max \left( 0, \max_{2k-1 \leq n \leq 2k} a[n] \right), \quad 1 \leq k \leq 5
\end{align*}
\]

Suppose you know the input \( x[n] \), and you know \( \epsilon[k] = \frac{\partial E}{\partial y[k]} \). Find \( \frac{\partial E}{\partial w} \) in terms of \( x[n] \) and \( \epsilon[k] \).
Problem 3  (16 points)

Remember that an affine transform is defined by a matrix with the following form:

\[
A = \begin{bmatrix}
a & b & c \\
d & e & f \\
0 & 0 & 1
\end{bmatrix}
\]

Define the scalar term \( \lambda \) to be \( \lambda = bd - (a - 1)(e - 1) \). It turns out that, as long as \( \lambda \neq 0 \), there is exactly one input vector of the form \( \vec{u}_0 = [u_0, v_0, 1]^T \) that maps to itself (\( A\vec{u}_0 = \vec{u}_0 \)). Find \( u_0 \) and \( v_0 \) in terms of \( a, b, c, d, e, f \) and \( \lambda \). HINT: you may find it useful to know that the inverse of a \( 2 \times 2 \) matrix is

\[
\begin{bmatrix}
\alpha & \beta \\
\gamma & \delta
\end{bmatrix}^{-1} = \frac{1}{\alpha\delta - \beta\gamma} \begin{bmatrix}
\delta & -\beta \\
-\gamma & \alpha
\end{bmatrix}
\]
Problem 4  (17 points)

Suppose that you have a training dataset with \( n \) training tokens \( \{(\vec{x}_1, \zeta_1), \ldots, (\vec{x}_n, \zeta_n)\} \), where \( \vec{x}_i = [x_{i1}, \ldots, x_{ip}]^T \), and \( \zeta_i \in \{0, 1\} \). You have a one-layer neural network that tries to approximate \( \zeta_i \) with \( z_i \), computed as \( z_i = \sigma(\vec{w}^T \vec{x}_i) \), where \( \sigma(\cdot) \) is the logistic function, and \( \vec{w} \) is a weight vector. Suppose that you want to maximize the accuracy of \( z_i \), but you also want to make \( \vec{w}^T \vec{x}_i \) as small as possible. One way to do this is by using a two-part error metric,

\[
E = -\frac{1}{n} \sum_{i=1}^{n} (\zeta_i \ln z_i + (1 - \zeta_i) \ln(1 - z_i)) + \frac{1}{2n} \sum_{i=1}^{n} (\vec{w}^T \vec{x}_i)^2
\]

Find \( \nabla_{\vec{w}} E \), the gradient of \( E \) with respect to \( \vec{w} \).
Problem 5  (17 points)

The Barycentric coordinates of point $\vec{x}_0 = [x_0, y_0, 1]^T$, as defined by the triangle $\vec{x}_1 = [x_1, y_1, 1]^T, \vec{x}_2 = [x_2, y_2, 1]^T, \vec{x}_3 = [x_3, y_3, 1]^T$, are the coordinates $\lambda_1, \lambda_2, \lambda_3$ such that $\vec{x}_0 = \lambda_1 \vec{x}_1 + \lambda_2 \vec{x}_2 + \lambda_3 \vec{x}_3$. If we constrain $\lambda_1 + \lambda_2 + \lambda_3 = 1$, then there are actually only two degrees of freedom; for example, we could substitute $\lambda_3 = 1 - \lambda_1 - \lambda_2$. A more interesting way to specify the two degrees of freedom is by defining variables $a$ and $b$ such that

$$\begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} ab \\ (1-a)b \\ 1-b \end{bmatrix}$$

Draw a two-dimensional Cartesian plane, and label the $x$ and $y$ axes. Label the point $\vec{x}_1 = [0, 0, 1]^T$, $\vec{x}_2 = [2, 0, 1]^T$, $\vec{x}_3 = [1, 2, 1]^T$, and $\vec{x}_0 = [1, 1, 1]^T$. Now, given the other four points, specify the line segment connecting the point $\vec{x}_3$ to the point $a\vec{x}_1 + (1-a)\vec{x}_2$. 
Problem 6  (17 points)

Consider a scalar LSTM, with a scalar memory cell, input gate, output gate, and forget gate, related to one another by scalar coefficients $a_i, b_i, a_o, b_o, a_f, b_f, a_c, b_c$ as follows:

\[
\begin{align*}
  i[n] &= \text{input gate} = \sigma(b_i x[n] + a_i c[n-1]), \quad 1 \leq n \\
  o[n] &= \text{output gate} = \sigma(b_o x[n] + a_o c[n-1]), \quad 1 \leq n \\
  f[n] &= \text{forget gate} = \sigma(b_f x[n] + a_f c[n-1]), \quad 1 \leq n \\
  c[n] &= f[n] c[n-1] + i[n] (b_c x[n] + a_c c[n-1]), 1 \leq n \\
  y[n] &= o[n] c[n], \quad 1 \leq n
\end{align*}
\]

Suppose that the network is initialized with $b_i = b_o = b_f = a_i = a_o = a_f = a_c = 0$, and $c[0] = 0$. In fact, the only nonzero coefficient is $b_c = 1$. Under this condition, find a formula for $y[n]$ in terms of the values of $x[m]$, $1 \leq m \leq n$. No variables other than $x[m]$ should appear in your answer. HINT: $\sigma(0) = 1/2$. 