ICSI 401 Final Exam (90 points)
November 30, 2020 Name: _____

This exam is open book, open note. Read all directions carefully and write your answers in the space provided (you can use additional space if you need it). To receive full credit, you must show all of your work. When you are finished, submit your scan, Word document, or PDF document to Blackboard.
The due date is Sunday, December 6, 2020, at 11:59 p.m.

1. (10 points) Calculate the Lagrange form of the interpolating polynomial for the following points: $(0, 1), (1, 3), (2, 12)$.

2. (10 points) Complete the divided differences table for the above points. Your table should be in the following form:

$f[0]$	X	X
$f[1]$	$f[0, 1]$	X
$f[2]$	$f[1, 2]$	$f[0, 1, 2]$

Here, an X indicates that there should NOT be a number in that entry.

3. (10 points) Consider the following divided differences table for the points $(0,1),(1,2),(2,4)$.

1		
2	1	
4	2	1/2

Use the divided differences table to write down the Newton form of the interpolating polynomial for the given points.

4. (10 points) Suppose that we wish to find a piecewise polynomial interpolation for the points $(x_0, y_0) = (0,1), (x_1, y_1) = (1,3), (x_2, y_2) = (2,1)$. In particular, we want two polynomials $P_0(x)$ and $P_1(x)$ satisfying various matching conditions, and we define

$$P(x) = \begin{cases} P_0(x) & x \in [x_0, x_1) \\ P_1(x) & x \in [x_1, x_2] \end{cases} \tag{1}$$

Write down a condition on the values of P_0 and P_1 that is *necessary* and *sufficient* to conclude that $P(x)$ is continuous on the interval $[x_0, x_2]$.

Hint: We already know that $P_0(x)$ and $P_1(x)$ are continuous on their respective intervals. So what you need in order to conclude that $P(x)$ is continuous on the whole interval is some matching condition.

5. (10 points) The following pseudocode implements Horner's rule for polynomial evaluation.

```
% Implements Horner's method for evaluating a polynomial at a point x.
%
% Inputs:
%  * coefficients: a row vector of polynomial coefficients, with the lowest-degree
% coefficient first.  For example, [1, 2, 3] corresponds to the polynomial
% p(x) = 1 + 2x + 3x^2.
%  * x: a floating point number.
%
% Output:
% The value of p(x).
%
function y = horner(coefficients, x)
    degree = size(coefficients)-1;
    degree = degree(2);

    if (degree == 1)
        y = x;
    else
        % degree > 1.
        z = horner(coefficients([2:degree]), x)
        y = coefficients(1) + x * z;
    end
end
```

Use Horner's rule to evaluate the polynomial $F(x) = x^3 + 3x^2 + x + 1$ at the point $x = 0$. In particular, you should fill in the missing values in the following table:

coefficients	z	y
[1, 1, 3, 1]		
[1, 3, 1]		
[3, 1]	0	3
[1]	N/A	0

NOTE: The above pseudocode is incorrect. The table is, as well. The following is correct (or should be, I hope):

```
% Implements Horner's method for evaluating a polynomial at a point x.
%
% Inputs:
%  * coefficients: a row vector of polynomial coefficients, with the lowest-degree
% coefficient first.  For example, [1, 2, 3] corresponds to the polynomial
% p(x) = 1 + 2x + 3x^2.
%  * x: a floating point number.
%
% Output:
% The value of p(x).
```

```
%
function y = horner(coefficients, x)
    degree = size(coefficients)-1;
    degree = degree(2);
    if (degree == 0)
        y = coefficients(1)
    else
        % degree > 1.
        z = horner(coefficients(2:(degree+1)), x)
        y = coefficients(1) + x * z;
    end
end
```

coefficients	z	y
[1, 1, 3, 1]		
[1, 3, 1]		
[3, 1]	1	3
[1]	N/A	1

Complete either table using the corresponding pseudocode for full credit.

6. (10 points) Power method

Consider the following matrix:

$$A = \begin{pmatrix} 2 & 4 \\ 4 & 3 \end{pmatrix} \tag{2}$$

Consider the following Matlab code for computing an eigenvector corresponding to the *second-largest* eigenvalue of A.

```
A = [2, 4; 4, 3];
%
% Code for computing an eigenvector corresponding to the second-largest
% eigenvector of A.
%
x = [1, 0]';
first_eigenvector = [0.661802563235740, 0.749678175815866]';
% Make x orthogonal to the first eigenvector by subtracting off its projection.
x = x - x' * first_eigenvector * first_eigenvector;
```

```
% Apply the power method for 1000 iterations.
for j = 1:1000
    % Line B
    x = A*x;
    x = x/norm(x);      % Line D
end
% Line C
display(x);
```

Which of the following modifications of the code will ensure convergence to the correct answer when run on a computer?

A. No modification is needed. It will converge to the correct answer.

B. At the beginning of the for loop (marked with "Line B"), insert the following code:

```
x = x - x' * first_eigenvector * first_eigenvector;
```

C. After the for loop (marked "Line C"), insert the following code:

```
x = (A' * A) \ (A' * x);
```

so as to solve the normal equations.

D. Remove the line marked by "Line D".

7. (10 points) Consider the *backward difference approximation* for the derivative of a function f:

$$\hat{f}'(x) = \frac{f(x) - f(x-h)}{h}, \tag{3}$$

for a small parameter $h > 0$.

Ignoring rounding error, what is the absolute error in approximating $f'(x)$ by $\hat{f}'(x)$, expressed in terms of h?

A. $|f'(x) - \hat{f}'(x)| = O(h)$ as $h \to 0$.

B. $|f'(x) - \hat{f}'(x)| = O(h^2)$ as $h \to 0$.

C. $|f'(x) - \hat{f}'(x)| = O(h^3)$ as $h \to 0$.

D. $|f'(x) - \hat{f}'(x)| = O(h^{-1})$ as $h \to 0$.

Hint: Use the Taylor series of f centered around x.

8. (10 points) Suppose that you wish to compute a least-squares fit to the following points using a polynomial $p(x)$ of degree ≤ 4: $(x_0, y_0) = (0, 1), (x_1, y_1) = (1, 3), (x_2, y_2) = (2, 4), (x_3, y_3) = (3, 2), (x_4, y_4) = (4, 5)$.

   What is the value of the approximation error of the least-squares degree ≤ 4 polynomial for these points (i.e., you need to calculate $\sum_{j=0}^{4}(y_j - p(x_j))^2$)?

   Hint: If you're doing a lot of calculating for this problem, then you're probably doing something wrong.

9. (10 points) Suppose that we want to approximate $f'(x)$, with $f(x) = \sin(x)$ and $x = \pi/3$, so that $f'(x) = 0.5$.

   We will use the forward difference formula:

$$\hat{f}'(x) = \frac{f(x+h) - f(x)}{h}. \tag{4}$$

   Below is a table giving the relative error of the approximation for various values of h, when implemented in Matlab.

h	$\left\lvert \frac{f'(x) - \hat{f}'(x)}{f'(x)} \right\rvert$
10^{-5}	8.660265948035038e-06
10^{-6}	8.660562260676128e-07
10^{-7}	8.601352896597801e-08
10^{-8}	6.077471192966753e-09
10^{-9}	8.274037077704576e-08
10^{-10}	8.274037077704576e-08
10^{-11}	8.274037077704576e-08
10^{-12}	8.890058234078956e-05
10^{-13}	7.992778373593354e-04
10^{-14}	7.992778373593354e-04
10^{-15}	0.110223024625156

   Notice, in particular, that the relative error decreases until $h \approx 10^{-8}$, and then it begins to increase as we further decrease h. What is the source of this phenomenon?

   A. The Runge phenomenon.

   B. Rounding error in the computation of $f(x+h)$ and $f(x)$.

   C. A bug in Matlab's source code.

   D. f is not differentiable near $x = \pi/3$.