1. If \( x_i, i = 0, 1, \ldots, n \) are distinct, we can fit a unique polynomial
\[
y = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} + a_n x^n
\]
through the data points \((x_0,y_0), (x_1,y_1), \ldots, (x_n,y_n)\). Suppose we for some reason were to try to fit the same data instead with a polynomial of the form
\[
y = b_0 + b_1 x + \ldots + b_{n-1} x^{n-1} + b_{n+1} x^{n+1}
\]
(i.e. \( x^{n+1} \) instead of \( x^n \) in the last term). Show that this must always fail (either have zero or infinitely many solutions) if the nodes happen to be distributed in such a way that \( x_0 + x_1 + \ldots + x_n = 0 \).

**Hint 1:** The special cases for \( n = 1, 2, 3 \) are illustrative:

\[
n = 1: \quad \det \begin{bmatrix} 1 & x_0^2 & x_0^4 \\ 1 & x_0^2 & x_0^4 \\ 1 & x_1 & x_2 \end{bmatrix} = (x_0 + x_1)(x_1 - x_0) = (x_0 + x_1) \prod_{0 \leq j < 1} (x_j - x_i)
\]

\[
n = 2: \quad \det \begin{bmatrix} 1 & x_0 & x_0^3 \\ 1 & x_1 & x_1^3 \\ 1 & x_2 & x_2^3 \end{bmatrix} = (x_0 + x_1 + x_2) \prod_{0 \leq j < 2} (x_i - x_j)
\]

\[
n = 3: \quad \det \begin{bmatrix} 1 & x_0 & x_0^2 & x_0^4 \\ 1 & x_1 & x_1^2 & x_1^4 \\ 1 & x_2 & x_2^2 & x_2^4 \\ 1 & x_3 & x_3^2 & x_3^4 \end{bmatrix} = (x_0 + x_1 + x_2 + x_3) \prod_{0 \leq j < 3} (x_i - x_j)
\]

Showing that this factorization pattern continues indefinitely will prove the result.

**Hint 2:** One somewhat different way to proceed is as follows: Write down the matrix that needs to be tested whether it is singular or not. Temporarily call \( x_0 \rightarrow x \) and note that the determinant becomes a polynomial in \( x \). Use then the relations between roots and coefficients for polynomials.

2. Using Matlab, generate a table of \( \Gamma(x) \) for \( x = 3.11, 3.12, \ldots, 3.17 \).

   a. Use then Stirling's interpolation formula to approximate \( \Gamma(\pi) \), and compare with the correct value.

   b. Use inverse interpolation to find a value \( x \) for which \( \Gamma(x) = 2.3 \).

3. The following table gives approximate values for smooth function, but it contains a typo. Correct it.

<table>
<thead>
<tr>
<th>( x )</th>
<th>0.0</th>
<th>0.1</th>
<th>0.2</th>
<th>0.3</th>
<th>0.4</th>
<th>0.5</th>
<th>0.6</th>
<th>0.7</th>
<th>0.8</th>
<th>0.9</th>
<th>1.0</th>
<th>1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(x) )</td>
<td>3.16</td>
<td>3.27</td>
<td>3.40</td>
<td>3.55</td>
<td>3.72</td>
<td>3.92</td>
<td>4.12</td>
<td>4.35</td>
<td>4.60</td>
<td>4.87</td>
<td>5.16</td>
<td>5.47</td>
</tr>
</tbody>
</table>

4. The cubic Hermite interpolation polynomial based on data for \( f(a), f'(a), f(b), f'(b) \) was stated in class (and is also given as equation (3.6.12) in the text book). Derive the same polynomial by instead using the standard Newton interpolation procedure (in its enhanced version that also allows the use of derivative information). Verify that the two procedures for obtaining the Hermite interpolation polynomial in fact produce the identical result (OK to use Mathematica).