1. Prove the first derivative recursion relations for orthogonal polynomials in the cases of:
   a. Chebyshev polynomials:
      \[ T_{n+1}' - \frac{T_n'}{n-1} = 2T_n. \]
   b. Legendre polynomials:
      \[ P_{n+1}' - \frac{P_n'}{n+1} = (2n+1)P_n. \]
   Hints: For (a), the easiest approach is probably to start with one of the explicit formulas for \( T_n(x) \).
                  For (b), you might try to proceed in the style of how the three-term recursions were demonstrated in class.

2. A Taylor expansion, centered at the origin, will converge within the largest surrounding circle in the complex plane inside which the function is singularity free. For a Chebyshev expansion, the region of convergence will similarly be the largest ellipse with focus points at +1 and -1. This is much more advantageous, since a singularity in the complex plane outside the real line segment [-1,1] then cannot cause divergence of the expansion over any part of this interval.

   An explicit example of the difference between the two types of expansions is provided by the function \( \arctan x \), for which both can be written down explicitly:
   \[ \arctan x = \sum_{k=0}^{\infty} \frac{(-1)^k}{2k+1} x^{2k+1} = \sum_{k=0}^{\infty} \frac{2(-1)^k}{2k+1} (\sqrt{2} - 1)^{2k+1} T_{2k+1}(x). \]

   Computationally compare the convergence rates of these two expansions by truncating both sums at some different choices for \( k = N \), and then plot their respective errors over [-1,1].

3. Chebyshev expansion coefficients are often not known analytically, but need to be computed numerically. The following very brief Matlab code uses the Fast Cosine Transform (implemented here using the standard \texttt{fft} routine rather than the \texttt{dct} routine in the Signal Processing toolbox) to approximate the leading coefficients in the Chebyshev expansion of an arbitrary function.

   \[
   \begin{align*}
   \text{N} &= \ldots ; \\
   \text{x} &= \cos(\pi \cdot [0:2\cdot\text{N}-1]/\text{N}) ; \\
   \text{a} &= \text{atan(x)} ; \\
   \text{f} &= \text{fft(a)}/\text{N} ; \quad \text{f(1) = f(1)/2} ; \\
   \text{real(f(1:12))} &= \text{Print out the first 12 coefficients}
   \end{align*}
   \]

   Run this code with \( N = 16 \) and \( N = 32 \) (specifying \texttt{format long}) and find out how many digits are correct in the resulting approximations (when comparing against the exact coefficients given in Problem 2 above).

4. In the ‘Orthogonal polynomials’ notes on the class web page, the Hermite polynomials \( H_n(x) \) are defined near the bottom of page 2, and given in explicit form through Rodrigues’ Formula (near the bottom of page 4). Defining \( H_n(x) \) by means of Rodrigues’ Formula:

   a. Determine explicitly \( H_0, H_1, H_2 \).

   b. Verify that \( H_n(x) \) satisfy the required orthogonality (including \( \int_{-\infty}^{\infty} H_n(x)^2 e^{-x^2} dx = \sqrt{\pi} 2^n n! \); you can assume it known that \( \int_{-\infty}^{\infty} e^{-x^2} dx = \sqrt{\pi} \)).

   c. Verify the differential equation for \( H_n(x) \) (also stated on page 4).

   d. Verify the three-term recursion relation (also stated on page 4).