1 (a) Find an exact formula for the cubic polynomial \( P_3(x) = x^3 + \cdots \) such that
\[
\int_{-1}^{1} P_3(x) q(x) dx = 0
\]
for any quadratic polynomial \( q \).

(b) Find exact formulas for the three roots \( x_1, x_2, x_3 \) of the equation \( P_3(x) = 0 \).

(c) Find exact formulas for the integration weights \( w_1, w_2, w_3 \) such that
\[
\int_{-1}^{1} q(x) dx = \sum_{j=1}^{3} w_j q(x_j)
\]
exactly whenever \( q \) is a polynomial of degree 5.

(d) Given any real numbers \( a < b \), find exact formulas for points \( y_j \in [a, b] \) and weights \( u_j > 0 \) such that
\[
\int_{a}^{b} q(x) dx = \sum_{j=1}^{3} u_j q(y_j)
\]
whenever \( q \) is a polynomial of degree 5.

(e) Explain why each of the three factors in the error estimate
\[
\int_{a}^{b} f(x) dx - \sum_{j=1}^{3} u_j f(y_j) = C_6 f^{(6)}(\xi) \int_{a}^{b} (y - y_1)^2 (y - y_2)^2 (y - y_3)^2 dy
\]
is inevitable and determine the exact value of the constant \( C_6 \).

(f) Use your code `ectr.m` to evaluate
\[
E_6 = \int_{-1}^{1} (x - x_1)^2 (x - x_2)^2 (x - x_3)^2 dx
\]
to 3-digit accuracy. Use parameters \( r = [x_1, x_2, x_3] \).

2 Implement, debug and test a MATLAB function `pleg.m` of the form

```matlab
function p = pleg(t, n)
    % t: evaluation point
    % n: degree of polynomial

    This function evaluates a single value \( P_n(t) \) of the monic Legendre polynomial \( P_n \) of degree \( n \), at evaluation point \( t \) with \(|t| \leq 1\). Here \( P_0 = 1, P_1(t) = t \) and \( P_n \) is determined by the recurrence
\[
P_n(t) = tP_{n-1}(t) - c_n P_{n-2}(t)
\]
for \( n \geq 2 \), where \( c_n = (n-1)^2/(4(n-1)^2 - 1) \). Be sure to iterate forward from \( n = 0 \) rather than recurse backward from \( n \), and do not generate any new function handles. Test that your function gives the right values for small \( n \) where you know \( P_n \).
Implement a MATLAB function `gaussint.m` of the form

```matlab
function [w, t] = gaussint(n)
% n: Number of Gauss weights and points
```

which computes weights \( w \) and points \( t \) for the \( n \)-point Gaussian integration rule

\[
\int_{-1}^{1} f(t) dt \approx \sum_{j=1}^{n} w_j f(t_j).
\]

(a) Find the points \( t_j \) to as high precision as possible, by applying your code `bisection.m` to `pleg.m`. Bracket each \( t_j \) initially by the observation that the zeroes of \( P_{n-1} \) separate the zeroes of \( P_n \) for every \( n \). Thus the single zero of \( P_1 = t \) separates the interval \([-1, 1]\) into two intervals, each containing exactly one zero of \( P_2 \). The two zeroes of \( P_2 \) separate the interval \([-1, 1]\) into three intervals, and so forth. Thus you will find all the zeroes of \( P_1, P_2, \ldots, P_{n-1} \) in the process of finding all the zeroes of \( P_n \).

(b) Find the weights \( w_j \) to as high precision as possible by applying your code `ectr.m` to

\[
w_j = \int_{-1}^{1} L_j(t)^2 dt
\]

where \( L_j \) is the \( j \)th Lagrange basis polynomial for interpolating at \( t_1, t_2, \ldots, t_n \).

(c) For \( 1 \leq n \leq 20 \), test that your weights and points integrate monomials \( f(t) = t^j \) exactly for \( 0 \leq j \leq 2n - 1 \).

4 (a) Show that

\[
\int_{0}^{1} x^{-x} dx = \sum_{n=1}^{\infty} n^{-n}
\]

(b) Use the sum in (a) to evaluate the integral in (a) to 12-digit accuracy.

(c) Evaluate the integral in (a) by `ectr.m` to 1, 2, and 3-digit accuracy. Estimate how many function evaluations will be required to achieve \( p \)-digit accuracy for \( 1 \leq p \leq 12 \). Explain the agreement or disagreement of your results with theory.

5 (a) Write, test and debug an adaptive 3-point Gaussian integration code `gadap.m` of the form

```matlab
function [int, abt] = gadap(a, b, f, r, tol)
% a,b: interval endpoints with a < b
% f: function handle f(x, r) to integrate
% r: parameters for f
% tol: User-provided tolerance for integral accuracy
% int: Approximation to the integral
% abt: Endpoints and approximations
```
Build a list $\mathbf{abt} = \{[a_1, b_1, t_1], \ldots, [a_n, b_n, t_n]\}$ of $n$ intervals $[a_j, b_j]$ and approximate integrals $t_j \approx \int_{a_j}^{b_j} f(x, r)dx$, computed with 3-point Gaussian integration. Initialize with $n = 1$ and $[a_1, b_1] = [a, b]$. At each step $j = 1, 2, \ldots$, subdivide interval $j$ into left and right half-intervals $l$ and $r$, and approximate the integrals $t_l$ and $t_r$ over each half-interval by 3-point Gaussian quadrature. If

$$|t_j - (t_l + t_r)| > \text{tol} \max(|t_j|, |t_l| + |t_r|)$$

add the half-intervals $l$ and $r$ and approximations $t_l$ and $t_r$ to the list. Otherwise, increment $\text{int}$ by $t_j$. Guard against infinite loops and floating-point issues as you see fit and briefly justify your design decisions in comments.

(b) Approximate the integral $\int_0^1 x^{-x}dx$ using your code from (a). Tabulate the total number of function evaluations required to obtain $p$-digit accuracy for $1 \leq p \leq 12$. Compare your results with the results and estimates for endpoint-corrected trapezoidal integration obtained in problem 4.