May 2nd, 2012 **Name (Please Print)** _____

CS II - Semesteral Exam- Semester II 11/12

Your Signature _____

Instructions:
Maximum time is 3 hours. Maximum possible score is 100.

Show all your work. Correct answers with insufficient or incorrect work will not get any credit.

Score

1.	(30)	
2.	(20)	
3.	(20)	
4.	(10)	
5.	(30)	
Total.	(110)	

Please attach this sheet on top of the first page of your answer script.

May 2nd, 2012 **Name (Please Print)** _____

CS II - Semesteral Exam- Semester II 11/12

Your Signature _____

1. (a) (15 points) A matrix $H = (h_{ij})$ is called upper Hessenberg if $h_{ij} = 0$ when $i > j + 1$. Describe the Gaussian elimination algorithm required to solve $Hx = f$, assuming you never need to swap rows How many flops (floating point operations) are needed if $H \in \mathbb{R}^{n\times n}$? (The answer should be put into the big O notation, i.e., of the type $O(n^k)$, with the smallest possible k.)

(b)(15 points) Manually perform three steps of Euler's method to solve

$$\frac{dy}{dt} = \frac{1}{t + y + 1},\ y(0) = 0$$

with $h = 0.2$.

2. (a) (10 points) Gauss quadrature on $[-1, 1]$ with three node points is given by

$$G_3(f) = \frac{5}{9}f(\frac{-\sqrt{15}}{5}) + af(0) + \frac{5}{9}f(\frac{\sqrt{15}}{5})$$

Determine the value of constant a and then use this quadrature formula to approximate

$$\int_0^1 (\ln x^2 + (2x - 1)^2)dx.$$

(b) (10 points) If we approximate the integral $\int_a^b f(x)dx$ by the composite trapezoidal rule with n subintervals, $T(n)$, we obtain

$$T(24) = 0.80326, \qquad T(48) = 0.80440, \qquad T(96) = 0.80468.$$

Use this information to compute the composite Simpson's rule estimates with n subintervals, $S(n)$, for $n = 24$ and $n = 48$.

3. (15 points) Let

$$A = \begin{bmatrix} 1 & -2 & 3 & 0 \\ 1 & -2 & 3 & 1 \\ 2 & 1 & 3 & -1 \\ 1 & -2 & 2 & -2 \end{bmatrix}.$$

Does A have an LU factorization where L is lower triangular with 1's on its diagonal and U is upper triangular? If not, determine if there is a permutation matrix P such that $PA = LU$. Find the matrix L, U and P.

4. (a) (10 points) For certain function $f(x)$, we know $f[0] = 1$, $f[0,1] = -1$, $f[0,1,2] = 2$. Furthermore, we know the absolute value of $f[0,1,2,x]$ is less than or equal to 3 for any $x \in [0,1]$. Determine the quadratic polynomial $p_2(x)$ that interpolates $f(x)$ at $x = 0, 1, 2$. Then find a good upper bound for $|f(0.5) - p_2(0.5)|$.

(b) (10 points) Determine the parameters a, b, c, d and e so that S is a cubic spline interpolation with natural end conditions.

$$S(x) = \begin{cases} a + b(x-1) + c(x-1)^2 + d(x-1)^3 & x \in [0,1] \\ (x-1)^3 + ex^2 - 1 & x \in [1,2] \end{cases}$$

5. (30 points) Please fill in the blanks so as to ensure that the below (three) program runs correctly in OCTAVE.
5(a)

```
 function r = bisect(fun,xb,xtol,ftol,verbose)
% bisect  Use bisection to find a root of the scalar equation f(x) = 0
%
% Synopsis:  r = bisect(fun,xb)
%            r = bisect(fun,xb,xtol)
%            r = bisect(fun,xb,xtol,ftol)
%            r = bisect(fun,xb,xtol,ftol,verbose)
%
% Input: fun     = (string) name of function for which roots are sought
%        xb      = vector of bracket endpoints. xleft = xb(1), xright = xb(2)
%        xtol    = (optional) relative x tolerance.    Default:  xtol=5*eps
%        ftol    = (optional) relative f(x) tolerance. Default:  ftol=5*eps
%        verbose = (optional) print switch. Default: verbose=0, no printing
%
% Output:  r = root (or singularity) of the function in xb(1) <= x <= xb(2)
if size(xb,1)>1, warning('Only first row of xb is used as bracket');  end
if nargin < ----(1)----,  xtol = 5*eps;  end
if nargin < ----(2)----,  ftol = 5*eps;  end
if nargin < ----(3)----,  verbose = 0;   end

xeps = max(xtol,5*eps);         %  Smallest tolerances are 5*eps
feps = max(ftol,5*eps);
a = xb(1,1); b = xb(1,2);       %  Use first row if xb is a matrix
xref = abs(b - a);              %  Use initial bracket in convergence test
fa = feval(fun,a);   fb = feval(fun,b);
fref = max([abs(fa) abs(fb)]);  %  Use max f in convergence test
if sign(fa)==-----(4)----           %  Verify sign change in the interval
  error(sprintf('Root not bracketed by [%f, %f]',a,b));
end
```

```
if verbose
  fprintf('\nBisection iterations for %s.m\n',fun);
  fprintf('   k       xm            fm\n');
end
k = 0;  maxit = 50;      %  Current and max number of iterations
while k < maxit
  k = k + 1;
  dx = b - a;
  xm = a + 0.5*dx;       %  Minimize roundoff in computing the midpoint
  fm = feval(----(5)----,xm);
  if verbose, fprintf('%4d  %12.4e  %12.4e\n',k,xm,fm); end

  if (abs(fm)/fref < feps) | (abs(dx)/xref < xeps) % True when root is found
    r = ----(6)----;  return;
  end

  if sign(fm)==sign(fa)
    a = ----(7)----;  fa = ----(8)----;    %  Root lies in interval [xm,b],
                                           %    replace a and fa
  else
    b = ----(9)----;  fb = ----(10)----;    %  Root lies in interval [a,xm],
                                           %  replace b and fb
  end
end
warning(sprintf('root not within tolerance after %d iterations\n',k));
```

5 (b)

```
function yhat = hermint(x,f,fp,xhat)
% hermint  Piecewise-cubic Hermite interpolation
%
% Synopsis: yhat = hermint(x,f,fp,xhat)
%
% Input:    x     = vector of independent variable values
%           f, fp = vectors of f(x) and f'(x)
%           xhat = (scalar or vector) x values where interpolant is evaluated
%
% Output:   yhat = scalar or vector value of cubic hermite interpolant at
%                  x = xhat.  size(yhat) = size(xhat)

n = length(x);
if     length(f)~=n,    error('x and f are not compatible');
```

```
elseif length(fp)~=n,   error('x and fp are not compatible');   end

% --- Construct coefficients of the piecewise interpolants
x = x(:);  xhat = xhat(:);    %  Convert to column vectors
f = f(:);  fp = fp(:);
dx = diff(x);                 %  Vector of x(i+1) - x(i) values
divdif = diff(f)./dx;         %  Vector of divided differences, f[x(i),x(i+1)]
a = ----(11)----;
b = ----(12)----;
c = ( ----(13)---- ) ./dx;
d = ( ----(14)---- )   ./dx.^2;

% --- Locate each xhat value in the x vector
i = zeros(size(xhat));   %  i is index into x such that x(i) <= xhat <= x(i+1)
for m=1:length(xhat)     %  For vector xhat: x( i(m) ) <= xhat(m) <= x( i(m)+1 )
  i(m) = binSearch(x,xhat(m));
end

% --- Nested, vectorized evaluation of the piecewise polynomials
xx = xhat - x(i);
yhat = a(i) + xx.*(b(i) + xx.*(c(i) + xx.*d(i)) );
```

5 (c)

```
function [c,R2] = linefit(x,y)
% linefit    Least-squares fit of data to y = c(1)*x + c(2)
%
% Synopsis:   c     = linefit(x,y)
%            [c,R2] = linefit(x,y)
%
% Input:   x,y = vectors of independent and dependent variables
%
% Output:  c  = vector of slope, c(1), and intercept, c(2) of least sq. line fit
%          R2 = (optional) coefficient of determination; 0 <= R2 <= 1
%               R2 close to 1 indicates a strong relationship between y and x
if length(y)~= length(x),  error('x and y are not compatible');  end

x = x(:);  y = y(:);    %  Make sure that x and y are column vectors
A = ----(15)----;  %  m-by-n matrix of overdetermined system
c = ----(16)----;      %  Solve normal equations
if nargout>1
  r = ----(17)----;
  R2 = ----(18)----;
end
```