MIDTERM: COMPUTER SCIENCE II

Date: **26th Feb 2016**

(1) (2+3+3+4+4+5+5=26 points) Write down the output of the following commands in octave.
   (a) w=linspace(1,3,11)'
   (b) A=diag(1:3:10).^2
   (c) A=diag(1:10:3).^2
   (d) B=(ones(2,4)-eye(2,4))*diag(1:4)*(ones(4,2)-eye(4.2))
   (e) p=[1 0 0 1]; roots(p)
   (f) A=[1 2;3 4];B=[1 0;0 1]; disp(A.*B); disp(A*B)
   (g) str=sprintf(´pi is close to %1.3f and eps is %1.0e´,pi,eps); disp(str)

(2) (4+4+8+8=24 points) Write down a command or a short code to achieve the following goals:
   (a) Display real part, imaginary part and conjugate of a complex number z.
   (b) Display the plot of the function $f(x) = sin(x) + e^x$ for x between $-\pi$ and π.
   (c) Write a function using $fprintf$ to display a table consisting of 3 columns and n rows whose entries are the values of the functions x, $e^x + sin(x)$ and $e^x - sin(x)$ for a given vector of length n as given below.

x	$e^x + sin(x)$	$e^x - sin(x)$
1	3.5598	1.8768
1.5	5.4792	3.4842
...	...	...

   (d) Write a code to evalutate cube root of 7 correct upto five places of decimal using Newton's method.

(3) (15 points) In the Octave code written on the attached sheet, add comments in the code to explain the lines of the code which end with a % symbol. Also write a synopsis for the code. (Remember to attach the sheet with your answer script!!)

(4) (25 points) Modify the algorithm of the secant method so that all the estimates lie in the given bracket. Write an Octave code for a function which approximates the roots of a function bracketted by a given vector using this new algorithm. Use relative convergence criterion.

(5) (10 points) Explain the use of global variables in an Octave function.

For Problem 3

```
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
%        maxit   = ..........(fill in the blanks)
%
% 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 < 3,  xtol = 5*eps;  end   %
if nargin < 4,  ftol = 5*eps;  end   %
if nargin < 5,  maxit = 50;   end   %

xeps = max(xtol,5*eps);         %
feps = max(ftol,5*eps);
if maxit < 3, maxit=3; end      %

a = xb(1,1); b = xb(1,2);       %
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)==sign(fb)           %
  error(sprintf('Root not bracketed by [%f, %f]',a,b));
end

k = 0;  maxit = 50;
while k < maxit          %
  k = k + 1;
  dx = b - a;
  xm = a + 0.5*dx;       %  Minimize roundoff in computing the midpoint
  fm = feval(fun,xm);

  if (abs(fm)/fref < feps) | (abs(dx)/xref < xeps) %
    r = xm;  return; %
  end
  if sign(fm)==sign(fa)
    a = xm;  fa = fm;    %
  else
    b = xm;  fb = fm;    %
  end
end
warning(sprintf('root not within tolerance after %d iterations\n',k));
```