Test values

```matlab
>> a=[3 1 4 7 5]
a =
   3  1  4  7  5
>> b=[3;1;4;7;5]
b =
   3
   1
   4
   7
   5
>> c=rand(3)*10
c =
   2.9741  6.5011  4.0007
   0.4916  9.8299  1.9879
   6.9318  5.5267  6.2520
>> d=[3 1 4 7 5;9 2 8 3 1]
d =
   3  1  4  7  5
   9  2  8  3  1
____________________________________________________Check the response of max to the various inputs____________________________________________________
```

```matlab
>> [amax,maxpos]=max(a)
amax =
    7
maxpos =
    4
>> [amax,maxpos]=max(b)
amax =
    7
maxpos =
    4
>> [amax,maxpos]=max(c)
amax =
    6.9318  9.8299  6.2520
maxpos =
    3  2  3
>> [amax,maxpos]=max(d)
amax =
    9  2  8  7  5
maxpos =
    2  2  2  1  1
```
Solutions

1. Given a vector A, report the position of the largest value in A.

```matlab
function pos=find_pos(A)
    % find the position of the largest value in an array
    i=1; maxA=A(1); imax=i; n=length(A);
    while i <= n
        if A(i)>maxA
            maxA=A(i);
            imax=i;
        end
        i=i+1;
    end
    pos=imax;
end
```

```matlab
>> find_pos(a)
ans =
   4
```

2. Given the approximation: \( \partial = \sum_{i=0}^{\infty} \frac{(-1)^i}{0.5i + 0.25} \), write a user defined function that calculates an approximate value for \( \partial \) that is accurate to at least four significant digits. Roughly how many terms must you use in order to achieve this level of accuracy? What about accuracy to six significant digits?

```matlab
function [p1,n]=cal_pi(err)
    % calculate pi using a series expansion
    i=0; oldpi=1; newpi=(-1)^i/(0.5*i+0.25);
    while abs(oldpi-newpi)>err
        i=i+1;
        oldpi=newpi;
        newpi=newpi+(-1)^i/(0.5*i+0.25);
    end
    p1=newpi; n=i;
end
```

```matlab
>> [p1,n]=cal_pi(0.0001)
p1 =
   3.1416
n =
   20000
```
3. The following iterative equation may be used to approximate the square root of a number:

\[
\sqrt{a} : \quad x_n = \frac{1}{2} \left( x_g + \frac{a}{x_g} \right)
\]

Method:
- Guess any \( x \) value, \( x_g \)
- Calculate a new \( x \) value, \( x_n \)
- Use \( x_n \) as the next guess, \( x_g \)
- Recalculate \( x_n \), and repeat…
- Answer, \( x_n \), should converge to \( \sqrt{a} \)

Write a user defined function that receives the value “a” as an input and returns the approximated square root. How many iterations must you perform in order to achieve reasonable accuracy?

```matlab
function [sq_root,n]=fsq_root(A)
    %approximate the square root of A
    xn=5; xg=1; err=0.0001; i=0;
    while abs(xn-xg)>err
        i=i+1;
        xg=xn;
        xn=0.5*(xg+A/xg);
    end
    sq_root=xn; n=i;

    >> fsq_root(5)
    ans =
    2.2361
```

4. Write a user defined function that works in the same manner as Matlab’s built-in `sum()` function for vectors.

```matlab
function s = vsum(A)
    %sum the vector A
    s=0.0; n=length(A);
    for i=1:n
        s=s+A(i);
    end

    Check
    >> vsum(a)
    ans =
    20
    >> sum(a)
    ans =
```
function s = fsum(A)
% write a function to reproduce the action of the sum function

[n,m]=size(A);
if n==1
    s(1)=0.0;
    for i = 1:m
        s(1)=s(1)+A(1,i);
    end
else
    for i = 1:m
        s(i)=0.0;
        for j = 1:n
            s(i)=s(i)+A(j,i);
        end
    end
end

Results of testing fsum

>> fsum(a)
ans =
    20
>> sum(a)
ans =
    20
>> fsum(b)
ans =
    20
>> sum(b)
ans =
    20
>> fsum(c)
ans =
    10.3975   21.8577   12.2406
>> sum(c)
ans =
    10.3975   21.8577   12.2406
>> fsum(d)
ans =
    12   3   12  10   6
>> sum(d)
ans =
    12     3    12    10     6

5. Write a user defined function that works in the same manner as Matlab’s built-in max() function for vectors.

function [amax,apos] = fmax(A)
    %find the maximum and the position of the maximum in A
    amax=A(1); n=length(A);
    for i=1:n
        if A(i)>amax
            amax=A(i);
            apos=i;
        end
    end
end

>> [amax,n]=fmax(a)
amax =
    7
n =
    4
>> [amax,n]=fmax(b)
amax =
    7
n =
    4