RECURSIVE FUNCTIONS

Parts of a Recursive Algorithm

All functions with a recursive algorithms must have the following:

1. Base Case (i.e., when to stop)
2. Work toward Base Case
3. Recursive Call (i.e., call itself)

The "work toward base case" is where we make the problem simpler. The recursive call, is where we use the same algorithm to solve a simpler version of the problem. The base case is the solution to the "simplest" possible problem (For example, the base case to adding a list of numbers would be if the list had only one number... thus the answer is just that number).

EXAMPLE #1: FACTORIAL RECURSIVE FUNCTION

To develop the recursive function to the right we used the following properties of the factorials:

(1) N! = N (N-1)!
N!=N(N-1)(N-2)!
N!=N(N-1)(N-2)(N-3)….1!
(2) If N=1, then N!=1
(3) If N=0, then N!=1

```
1. function f = ufact(x)
2. % Computes the factorial of x
3.     if (x==0)
4.         f = 1;
5.     else
6.         f = x*ufact(x-1);
7.     end
8. end
```

Note how the function ufact call itself in line 8. Calling itself is the particular behavior for classification as “recursive” function

Let’s observe the mechanism of the ufact function. Assume a call to this function with x=5:

EXAMPLE #2. FIBONACCI SERIES RECURSIVE FUNCTION

The recursive formulation for the Fibonacci series, starts with F_1=0, F_2=1, and for i=3,4,5…n:

$$F_i = F_{i-1} + F_{i-2}$$

The sequence is F=[0,1,1,2,3,5,8,13,21,34……]; Each new term is the addition of the two previous ones. The first two terms of the series, i.e., F_1=0, F_2=1 can't be computed, they are definitions.

The recursive Fibonacci function ("uFib"):

```
function  [R] = uFib(n)
% Computes the n fibonacci number with
% fib(1)=0; fib(2)=1; as starting values
% must be n>=1

if n==1
      R=0;          % Stops the recursion
elseif n==2
      R=1;          % Stops the recursion
else
   R=uFib(n-1)+uFib(n-2);  % function call itself
end
end
```

Command Window:

```
>> n = 1;
>> uFib(n)
ans =   0

>> n = 2;
>> uFib(n)
ans =   1

>> n = 3;
>> uFib(n)
ans =   1

>> n = 4;
>> uFib(n)
ans =   2

>> n = 5;
>> uFib(n)
ans =   3
```

EXAMPLE #3. SUM OF CONSECUTIVE-NUMBERS RECURSIVE FUNCTION

A function that sums the natural number from 1 up to N. Sum is commutative, order the number backwards so the last one can become the 'base case'

$$N + N - 1 + \cdots + 5 + 4 + 3 + 2 + 1$$

The recursive algorithm:

$$\sum_{i=1}^{N} i = N + \sum_{i=1}^{N-1} i = N + (N-1) + \sum_{i=1}^{N-2} i =$$

$$\sum_{i=1}^{N} i = N + (N-1) + (N-2) + \sum_{i=2}^{N-3} i + 1$$

```
function [R]=uSum2(N)
% Computes the sum of integers i=1,2,3..,N

   if N==1
      R= 1;              % Stops the recursion
   else
      R= N+uSum2(N-1);   % function calls itself
   end
end
```

Command Window:

```
>> N=1; uSum2(N)
ans =
    1

>> N=2; uSum2(N)
ans =
    3

>> N=3; uSum2(N)
ans =
    6

>> N=4; uSum2(N)
ans =
   10
```

EXAMPLE #4. POWER-OF-A-NUMBER RECURSIVE FUNCTION

To develop the recursive function **uPowers** for the n power of X, i.e, X^n, we can express similarly

$$X^n = X\ X^{n-1}$$
$$X^n = X * X * X^{n-2}$$
$$X^n = X * X * X * X^{n-3}$$
$$\ldots$$
$$X^n = X * X * X * \ldots * X^1$$

The above expression work for n=1, 2, etc. It doesn't work for n=0.

Note that when n=1 then $X^n = X$ and when n=0 then $X^n=1$. Both of these results can be used for stopping the recursive calling.

```
function [Y]=uPower(x,n)
% Computes the n power of x number

if n==0
    Y = 1;
elseif n==1
    Y = x;                          % Stops recursion
else
    Y = x * uPower(x,n-1);          % function call itself
end
end
```

```
Command Window:
>> x=2;n=0;
>> uPower(x,n)
ans =   1

>> x=2;n=1;
>> uPower(x,n)
ans =   2

>> x=2;n=2;
>> uPower(x,n)
ans =   4

>> x=2;n=3;
>> uPower(x,n)
ans =   8
```

EXAMPLE #5. SUM-OF-A-CUBIC POWERS RECURSIVE FUNCTION

A function that sums the cubic powers of natural number up to n

$$1^3 + 2^3 + 3^3 + 4^3 + 5^3 + \cdots . + N^3$$

The recursive algorithm:

$$\sum_{i=1}^{N} i^3 = N^3 + \sum_{i=1}^{N-1} i^3 = N^3 + (N-1)^3 + \sum_{i=1}^{N-2} i^3 = N^3 + (N-1)^3 + (N-2)^3 + \sum_{i=2}^{N-3} i^3 + \cdots + 1^3 = \cdots$$

A function that sums the cubic powers of natural number up to n

```
function [R]=uCSum(N)
% Computes the sum of cubic powers of 1,2,3..,N numbers

if N==1
    R=1;                        % Stops the recursion
else
    R= N^3 + uCSum(N-1);        % function call itself
end
end
```

Command Window:

```
>> N=1; uCSum(N)
ans =
     1
>> N=2; uCSum(N)
ans =
     9
>> N=3; uCSum(N)
ans =
    36
>> N=4; uCSum(N)
ans =
   100
>> N=5; uCSum(N)
ans =
   225
```

EXAMPLE #6. ADD-NUMBERS RECURSIVE FUNCTION

A function that sums four numbers:

$$S = a + b + c + d$$

Can be expressed also as

$$\sum_{i=a,b,c,d} i = d + \sum_{i=a,b,c} i = d + c + \sum_{i=a,b} i$$

```
function [R]=uSum(a,b,c,d)
% Computes the sum of a, b, c, and d numbers

   if nargin()==2
      R= a+b;                  % Stops the recursion
   elseif nargin()==3
      R= uSum(a+b,c);          % function call itself
   elseif nargin()==4
      R= uSum(a+b,c,d);        % function call itself
   end
end
```

```
Command Window:
Trial>> a=1; b=2; c=3; d=4

Trial>> uSum(a,b,c,d)

ans =

   10
```

NOTE: The function nargin() counts the number of input arguments

EXERCISE #7. SUM OF FACTORIALS RECURSIVE FUNCTION

A function that sums the factorial of the natural numbers from 1 up to N

$$1! + 2! + 3! + 4! + 5! + \cdots . + N!$$

The recursive algorithm:

$$\sum_{i=1}^{N} i\,! = N! + \sum_{i=1}^{N-1} i! = N! + (N-1)! + \sum_{i=1}^{N-2} i! = N! + (N-1)! + (N-2)! + \sum_{i=1}^{N-3} i! = N! + (N-1)! + (N-2)! + \cdots + 1!$$

```
function [R]=sumFactorial(N)
% Computes the sum of factorials i=1,2,3..,N

    if N==1
        R= 1;                   % Stops the recursion
    else
        R= factorial(N)+sumFactorial(N-1);
        % function calls itself
    end
end
```

Command Window:

```
>> sumFactorial(3)
ans =
     9
>> sumFactorial(4)

ans =
    33
```

EXERCISE #8 The ppii(N) recursive function:

$$ppii(N) = 4\sum_{n=1}^{N} \frac{(-1)^{n+1}}{2n-1} = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \frac{4}{13} \ldots = 4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} \ldots)$$

Where n=1,2,3,4,…N. The recursive formula:

$$ppii(N) = 4\sum_{n=1}^{N}\frac{(-1)^{n+1}}{2n-1} = \frac{4(-1)^{N+1}}{2N-1} + 4\sum_{n=1}^{N-1}\frac{(-1)^{n+1}}{2n-1} = \frac{4(-1)^{N+1}}{2N-1} + \frac{4(-1)^{(N-1)+1}}{2(N-1)-1} + 4\sum_{n=2}^{N-2}\frac{(-1)^{n+1}}{2n-1} + \cdots + \frac{4}{1}$$

```
function [R] = ppii(N)
% Computes pi series with up to the N term

    if N==1
        R=4/1;
    else
        R=(((-1)^(N+1))*4)/(2*N-1)+ppii(N-1);
    end

end
```

```
>> ppii(4)

ans =

   2.8952

>> ppii(5)

ans =

   3.3397
```

EXAMPLE #9. SUM-OF-A-NATURAL LOGARITHMS OF NATURAL NUMBERS RECURSIVE FUNCTION

A function that sums the natural logarithms of natural number, n=1,2,3,4,… N:

$$\ln(1) + \ln(2) + \ln(3) + \ln(4) + \ln(5) + \cdots . + \ln(N)$$

The recursive algorithm:

$$\sum_{i=1}^{N}\ln(i) = \ln(N) + \sum_{i=1}^{N-1}\ln(i) = \ln(N) + \ln(N-1) + \sum_{i=1}^{N-2}\ln(i) = \ln(N) + \ln(N-1) + \sum_{i=2}^{N-2}\ln(i) + \cdots + \ln(1)$$

	Command Window:

```
function [R]=sumLOG(N)
% Computes the sum of natural logarithms of natural numbers
1,2,3..,N

if N==1
    R=log(N);                    % Stops the recursion
                                 % Also R=log(1); also R=0;
else
    R=log(N)+ sumLOG(N-1);       % function call itself
end
end
```

```
>> sumLOG(1)
ans =
     0
>> sumLOG(2)
ans =
    0.6931
>> sumLOG(3)
ans =
    1.7918
 >> sumLOG(4)
ans =
    3.1781
```

EXAMPLE #11. SUM OF INDEXED-NUMBERS RECURSIVE FUNCTION

A function that sums an indexed list of number from X(1), X(2),… up to X(n)

$$X(1) + X(2) + X(3) + \cdots + X(n)$$

An example of a list of numbers: x=[37, 13, 4, 12, 21, 31, 7, 10, 45]; (test your function with these numbers)

The recursive algorithm:

$$\sum_{i=1}^{n} X(i) = X(n) + \sum_{i=1}^{n-1} X(i) =$$

$$= X(n) + X(n-1) + \sum_{i=1}^{n-2} X(i) =$$

$$= X(n) + X(n-1) + X(n-2) + \cdots + \sum_{i=2}^{n-3} X(i) + \cdots X(1) =$$

```
function [R]=uSum4(X)
% Computes the sum of integers
% i=1,2,3..,N

n=numel(X);
 if n==1
   R= X(n); % Stops the recursion
 else
   R= X(n)+uSum4(X(1:n-1));
          % function calls itself
 end
end
```

Command Window:
>> x=[37, 13, 4, 12, 21, 31, 7, 10, 45];

>> SSSS=uSum4(x);
>> fprintf('The sum of the number list is %d \n', SSSS);

OUTPUT:
The sum of the number list is 180

EXAMPLE 10.

~~TRY the following exercise~~

$$PPii = \sqrt{6(1+\frac{1}{2^2}+\frac{1}{3^2}+\cdots)}$$

~~La solucion a continuacion NO TRABAJA, pero ensena detalles de como trabajan las funciones recursivas~~

```
function RR=ppii(x)
% Computes mathematical pi
   flag=0;
```

```
    if x==1
        RR=1;
        flag=1;
    else
        if flag==1
            RR=sqrt(6*(1/x^2)+ppii(x-1));    % must be executed only one time at the end
        else
            RR=(1/x^2)+ppii(x-1);
        end
    end
end
```