Linear Search and Binary Search : Recursive Versions

```
#-------------------------------------------------------------------------------

# Linear Search of a list with  n  items :
#
#    number of list items inspected ( worst case )  =  n .

#-------------------------------------------------------------------------------

def LinearSearch( item, lst ) :

    # A position in 'lst' in which 'item' occurs, or 'None' if not present

    return LinearSearchFrom( item, lst, 0 )

#-------------------------------------------------------------------------------

def LinearSearchFrom( item, lst, start ) :

    # A position in 'lst', from position 'start' onwards,
    # in which 'item' occurs, or 'None' if not present

    if start == len( lst ) :
        return None
    elif item == lst[ start ] :
        return start
    else :
        return LinearSearchFrom( item, lst, start + 1 )

#-------------------------------------------------------------------------------
```

```
#-------------------------------------------------------------------------------

>>> #      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
>>> L1 = [ 53, 76, 31, 12, 57, 85, 73, 49, 44, 60, 93, 18, 65, 96, 23 ]

>>> LinearSearch( 73, L1 )
6

>>> LinearSearch( 59, L1 )
None

#-------------------------------------------------------------------------------
```

```
#-------------------------------------------------------------------------------

# Binary Search of a list with  n  items :
#
#    number of list items inspected ( worst case )  =  log n [base 2]   (approx)

#-------------------------------------------------------------------------------

def BinarySearch( item, lst ) :

    # A position in 'lst' in which 'item' occurs, or 'None' if not present
    #
    # NOTE : 'lst' must already be sorted in ascending order

    return BinarySearchBetween( item, lst, 0, len( lst ) - 1 )

#-------------------------------------------------------------------------------

def BinarySearchBetween( item, lst, lo, hi ) :

    # A position in 'lst', between positions 'lo' and 'hi', inclusive,
    # in which 'item' occurs, or 'None' if not present
    #
    # NOTE : this section of 'lst' must already be sorted in ascending order

    if lo > hi :
        return None
    else :
        mid = ( lo + hi ) // 2
        if   item < lst[ mid ] :
            return BinarySearchBetween( item, lst, lo, mid - 1 )
        elif item > lst[ mid ] :
            return BinarySearchBetween( item, lst, mid + 1, hi )
        else :
            return mid

#-------------------------------------------------------------------------------
```

```
#-------------------------------------------------------------------------------

>>> #      0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
>>> L2 = [ 12, 18, 23, 31, 44, 49, 53, 57, 60, 65, 73, 76, 85, 93, 96 ]

>>> BinarySearch( 73, L2 )
10

>>> BinarySearch( 59, L2 )
None

#-------------------------------------------------------------------------------
```

Linear Search and Binary Search : Recursive Versions

```
#-----------------------------------------------------------------------------

# Linear Search of a list with  n  items :
#
#    number of list items inspected ( worst case )  =  n .

#-----------------------------------------------------------------------------

def LinearSearch( item, lst, start = 0 ) :

    # A position in 'lst', from position 'start' onwards,
    # in which 'item' occurs, or 'None' if not present

    if start == len( lst ) :
        return None
    elif item == lst[ start ] :
        return start
    else :
        return LinearSearch( item, lst, start + 1 )

#-----------------------------------------------------------------------------
```

```
#-----------------------------------------------------------------------------

>>> #       0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
>>> L1 = [ 53, 76, 31, 12, 57, 85, 73, 49, 44, 60, 93, 18, 65, 96, 23 ]

>>> LinearSearch( 73, L1 )
6

>>> LinearSearch( 59, L1 )
None
#-----------------------------------------------------------------------------
```

```
#-----------------------------------------------------------------------------

# Binary Search of a list with  n  items :
#
#    number of list items inspected ( worst case )  =  log n [base 2]  (approx)

#-----------------------------------------------------------------------------

def BinarySearch( item, lst, lo = 0, hi = None ) :

    # A position in 'lst', between positions 'lo' and 'hi', inclusive,
    # in which 'item' occurs, or 'None' if not present
    # ( take  None  to mean  len( lst ) - 1  for 'hi' )
    #
    # NOTE : this section of 'lst' must already be sorted in ascending order

    if hi == None :
        hi = len( lst ) - 1

    if lo > hi :
        return None
    else :
        mid = ( lo + hi ) // 2
        if  item < lst[ mid ] :
            return BinarySearch( item, lst, lo, mid - 1 )
        elif item > lst[ mid ] :
            return BinarySearch( item, lst, mid + 1, hi )
        else :
            return mid

#-----------------------------------------------------------------------------
```

```
#-----------------------------------------------------------------------------

>>> #       0   1   2   3   4   5   6   7   8   9  10  11  12  13  14
>>> L2 = [ 12, 18, 23, 31, 44, 49, 53, 57, 60, 65, 73, 76, 85, 93, 96 ]

>>> BinarySearch( 73, L2 )
10

>>> BinarySearch( 59, L2 )
None
#-----------------------------------------------------------------------------
```