Binary Search

Assume A is an array of length L, the elements are hence in locations 0 .. (L-1 )

We give you two binary search algorithms.

The first, binary_search_A, returns the index of the element if found in the array, else it returns (-1) to designate not found.

The second, binary_search_B, also returns the element index if found, but returns the index of about where the data SHOULD be if it had been found. This second method is useful when doing a range search and you want to get a starting point for the range search. From this starting point, one then needs to go left (lower) until the array elements have values less than the search range, and go right (greater) until arral elements have values greater than the search range.

-----------------------------

function binary_search_A (searchKey:Number) : Number
// this binary search returns the index of the number
// if found, else -1 to designate not found
{
    low = 0
    high = L-1 ;

    while( low <= high )
    {
        mid = Floor ( (low+high) / 2 )
        if (searchKey == A[mid])
            return (mid)
        else
        {
            if (searchKey < A[mid]) // need to look left of mid
                high = mid - 1
            else
                low = mid + 1
        }
    }
    return (-1) // not found
}
function binary_search_B (searchKey:Number) : Number
// this binary search returns the index of the number
// if found, else returns the index of about where the element
// should be if it were to be in the array
{
    low = 0
    high = L-1 ;

    while( low <= high )
    {
        mid = Floor ( (low+high) / 2 )
        if (searchKey == A[mid])
            return (mid)
        else
            {
                if (searchKey < A[mid])  // need to look left of mid
                    high = mid - 1
                else
                    low = mid + 1
            }
    }
    return (mid) // if got here, searchKey not in array, this is the index
                  // of about where it would be if it HAD been in the array