UCS 1312 Data Structures Lab
A10: Searching and Sorting - Applications

--Dr. R. Kanchana

Best Practices to be adapted

Modular design and coding using versions
Simple and Understandable UI design
Test case design – minimal and adequate test cases
Guiding error messages in case of invalid inputs
^^^

Write algorithms for applications. Design proper adequate test cases with expected output.

(CO4, K3)

1. Given an array of sorted integers with duplicate elements, write efficient C function
   getfirstOccur(a[ ], size, target) that returns the index of the first occurrence of target
   element in the array
   getlastOccur(a[ ], size, target) that returns the index of the last occurrence of target
   element in the array
   Example: A[ ] = {2, 34, 45, 47, 53, 53, 53, 53, 64, 64, 76, 89, 97}
   getfirstOccur(A[ ], 13, 64) returns 8
   getlastoccur(A[ ], 13, 53) returns 7
   getfirstOccur(A[ ], 13, 18)   returns -1 (Not found)

2. Write an efficient function in C CountOnes(a[], size) that counts the number of 1's in a
   sorted binary array.
   Example: a[ ] ={ 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1}
   CountOnes(a[ ], 14) returns 6.

3. You are given an array a, size of the array N. Follow selection sorting method and print
   the state of the array after each iteration has been performed.

   In your notebook, design atleast 3 test cases to realize best/worst/average cases. For
   each test case, trace the sorting method and write the state of the array in each
   iteration. Verify your expected output with the actual output of your function.

   ****