Assignment 2 (DS)

Q1. Write a C/C++ program to implement Linear Search. We need to check for duplicate inputs. If the duplicate element is allowed then a linear search algo needs to implement accordingly. An appropriate condition check needs to apply for element not found scenario.

(a) Search Element (iterative approach)

(b) Search Element (recursion approach)

(c) Print Input array (iterative approach)

(d) Print Input array (recursion approach)

(e) Print Input array reverse order (iterative approach)

(f) Print Input array reverse order (recursion approach)

Q2. Write a C/C++ program to implement Binary Search. We need to check for duplicate element inputs, if found any should not insert into the input array. The array should manage in sorted order. Apply a duplicacy check on the element while inserting the element, if found duplicate then discards the input. The input element should insert in the array at the right position(index). Appropriate shifting can be applied in the array in order to insert the input element at the right position. An appropriate condition check needs to apply for element not found scenario.

(a) Binary Search (iterative approach)

(b) Binary Search (recursion approach)

Q3. Find the Factor

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number) and then return the p^{th} element of the list, sorted ascending. If there is no p^{th} element, return *0*.

Example: *n = 20 and p = 3*

The factors of *20* in ascending order are *{1, 2, 4, 5, 10, 20}*. Using 1-based indexing, if *p = 3*, then *4* is returned. If *p > 6*, *0* would be returned.

Function Description

Complete the function *pthFactor* in the editor below.

pthFactor has the following parameter(s):

int n: the integer whose factors are to be found

int p: the index of the factor to be returned

Returns:

int: the long integer value of the p^{th} integer factor of *n* or, if there is no factor at that index, then *0* is returned

Q4. Count Duplicate Elements

Given an integer array, *numbers,* count the number of elements that occur more than once.

Example *numbers = [1, 3, 3, 4, 4, 4]*

There are two non-unique elements: *3* and *4.*

Function Description:

Complete the function *countDuplicate* in the editor below.

countDuplicate has the following parameter(s):

int numbers[n]: an array of integers

Returns:

int: an integer that denotes the number of non-unique values in the *numbers* array