DATA STRUCTURES
ASSIGNMENT - V
(FOR THE SEVENTH LAB SESSION)

Assignments to be completed during lab session

1. Perform bubble sort on an array of real values, such that, the best case time complexity of the sort is $O(N^2)$. Here, the size of the array, i.e., N and the elements of the array are user input. You need to dynamically allocate memory for the array.
2. Now, modify the program of bubble sort using a flag, such that, the best case time complexity reduces to $O(N)$.
3. Perform selection sort on an array of real values. The size of the array and the elements of the array need to be user input. You need to dynamically allocate memory for the array.
4. Perform insertion sort on an array of real values. The size of the array and the elements of the array need to be user input. You need to dynamically allocate memory for the array. You may skip this assignment if you have already done this in an earlier assignment.
5. Perform merge sort on an array of real values. The size of the array and the elements of the array need to be user input. You need to dynamically allocate memory for the array and any temporary array if required.
6. Perform quicksort on an array of real values. The size of the array N and the elements of the array need to be user input. You need to dynamically allocate memory for the array.
7. Write a function that returns the index of the last occurrence of an item in an array. If the element is not present. The function needs to return -1.
8. An array is sorted in descending order. Write a function to perform a binary search on the array. The function should return an index of the searched item if the item is present and -1 otherwise.
9. There are N elements in an array such that the first N_1 elements are sorted in descending order and the remaining $(N - N_1)$ elements are sorted in ascending order. Write a function to sort the array with $O(N)$ time complexity. Note that, N and the elements of the array are user input but N_1 is not user input.

10. There are N elements in an array such that the first N_1 elements are sorted in ascending order and the remaining $(N - N_1)$ elements are sorted in descending order. Write a function to sort the array with $O(N)$ time complexity. Note that, N and the elements of the array are user input but N_1 is not user input.

Additional assignments

1. Consider the following structure.

```
struct student_t {
    int    roll;  /* The roll no of the student.                  */
    char  *name;  /* A pointer to the name of the student.        */
    float  sgpa;  /* Semester grade point average of the student. */
};
```

   Create an array of `student_t`. Write a single function to sort the array either in terms of `roll` or `marks`. The function should be generic so that if later we want to sort the array in terms of `name`, we should be able to perform it without modifying the code of the function.

2. Write a function that can sort an array of any structure concerning any of its member variables.
   Hints: Input to the function includes (i) a pointer to the array (either in the form of `char *` or `void *`), (ii) the size of each element of the array, i.e., the size of the structure (in the form of `const unsigned int`), (iii) the size of the array, i.e., the number of elements in the array (in the form of `const unsigned int`), and (iv) a pointer to a comparison function that takes either two `void` or `char` pointers as arguments and returns an integer.

3. Write a function to sort a linked list using quicksort.
   Hints: Consider the first element in the list as the pivot element. Remove all elements from the list. Add each element in a linked list (say `smaller_list`) if the item is smaller than the pivot. Otherwise, add it in another linked list (say `larger_list`). Recursively sort the lists `smaller_list` and `larger_list`, such that, the terminating condition be a single element or an empty list. Now, merge the sorted `smaller_list`, the pivot item, and the sorted `larger_list` to get the sorted list.