Assignment 1-5

Notes:

(a) **If you are intelligent enough, you can avoid repetitive code and write a group of function in a single program.**

(b) **Try to write efficient code (less execution time and/or less space)**

(c) **Due date:** Just After I mid term. CR will be responsible to collect the hard copies and submit to me with the list who have submitted (no soft copies will be allowed).

Questions

1. Write a C program which read 10 integers from user and print all pair whose sum is 70.
2. Write a C program to read marks of 50 students ranged between 0 and 10 and find frequency of each mark.
3. Write a C program to calculate mean, variance and standard deviation of sorted n floating point values.
   - int **arr_mean** (int a[ ] , int n)
   - int **arr_varience** (int a[ ] , int n)
   - int **std_dev** (int a[ ] , int n)
4. Write a C program which reads a list of N numbers and finds the largest and second largest of them. User needs to write a function **arr_largest()** which accepts two arguments (array and no. of values in the array) and return largest value. Similarly, provide **arr_s_largest( )**.
   - int **arr_largest** (int a[ ] , int n)
   - int **arr_s_largest** (int a[ ] , int n)
5. Write a C program to perform linear search in an array of *n* elements. User needs to write a function **linear_search( )** which accepts three arguments (array , no. of values, and value to be searched in the array) and return its position, if value is found otherwise return -1.
   - int **linear_search** (int a[ ] , int n, int val)
6. Write a C program to count no. of occurrences of a particular in an array of *n* elements. User needs to write a function **linear_search_1( )** which accepts three arguments (array, no. of values, and value to be searched in the array) and return the count.
   - int **linear_search_1** (int a[ ] , int n, int val)
7. Write a C program to perform binary search in a sorted array. User needs to write a function **binary_search()**
   - int **binary_search** (int a[ ] , int val , int low_index, int high_index)

   which return its position, if value is found otherwise return -1. low_index and high_index are lower and upper bound of array, respectively.
8. Write a C program to sort an array of *n* elements using bubble sort. User needs to write a function **bubble_sort( )** which accepts two arguments (array and no. of values) and return nothing.

- void **bubble_sort** (int a[ ] , int n)

9. Repeat exercise 8 for selection sort.
10. Repeat exercise 8 for insertion sort.
11. Write a C program to sort an array of *n* elements in such a way that all even values precedes odd values.
    - void **sort_1** (int a[ ] , int n)
12. Write a C program to sort the first half of an array in the ascending order and the other half of the array in the descending order.
    - void **sort_2** (int a[ ] , int n)
13. Write a C program to merge two sorted arrays and store into 3 array.
    - void **sort_1** (int a[ ] , int m, int b[ ], , int n, int c [])
14. Write a C program to insert a value *val* at position *p* in an array of *n* elements.
    - void **arr_insert** (int a[ ] , int n, int val, int p)
15. Write a C program to insert a value *val* into a sorted array of *n* elements. (Hint: first search value using binary search then insert the value.)
    - void **arr_insert_1** (int a[ ] , int n, int val)
16. Write a C program to delete a value *val* (first occurrence) from an array of *n* elements.
    - void **arr_del** (int a[ ] , int n, int val)
17. Write a C program to delete a value from position *p* in an array of *n* elements.
    - void **arr_del_1** (int a[ ] , int n, int p)
18. Write a C program to count all duplicate elements in an array of *n* elements.
    - int **arr_dup_count** (int a[ ] , int n)
19. Write a C program to delete all duplicate elements in an array of *n* elements.
    - void **arr_dup_del** (int a[ ] , int n)