BIRLA INSTTUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJ.)
CS F111 Computer Programming
LAB SESSION #7
(Arrays)

Create a directory "**lab7**" inside "**myprogs**" directory for all your programs in this week's lab. We will write programs related to functions and arrays.

Arrays in C:
We have also studied arrays in C, how they are stored in memory and how they are processed. We have also seen how arrays are passed into functions using pass-by-reference and how the changes made to the array in a called function are also reflected in the array in the calling function. We have also learned about linear search, binary search, and selection sort. Refer to the slides to revise them. Now, let us try to implement them.

1. Write a program that takes 10 floating point numbers from the user as input and stores them in an **float array**, and computes the maximum element, second maximum element, minimum element, and second minimum element of an array. Then it computes the average of the four elements computed above and displays it to the user.

2. Create a file "**searchNSort.c**" in the "**myprogs**" directory. This file should contain four functions as follows:
   a. **linearSearch()** that takes an integer array **`arr`** and an integer element **`x`** as input and searches for **`x`** in the **`arr`** using ***linear search***. If **`x`** is found, it should return the position of **`x`** in **`arr`**. If **`x`** is not found, it should return **`-1`**. This function should not print anything.
   b. **binarySearch()** that takes an integer array **`arr`** and an integer element **`x`** as input and searches for **`x`** in the **`arr`** using ***binary search***. If **`x`** is found, it should return the position of **`x`** in arr. If **`x`** is not found, it should return **`-1`**. This function should not print anything.
   c. **selectionSortDec()** that taken an integer array **`arr`** as input and sorts it in ***decreasing order***. This function should not return or print anything.
   d. **main()** function that should display a prompt to the user to enter an option using an infinite loop. The prompt should display: *"Enter 1 for linear Search, 2 for Binary Search, 3 for Selection Sort, 4 to exit"*.
      i. If user enters 1 or 2, you should take an input of 10 integer elements from the user and store it in an array. You should also take an integer element to be searched in that array. Then call linearSearch() or binarySearch() depending upon user input. After the function returns, you must either print – "*Element found at i^{th} location*" or "*Element not found*". "*i*" is the rank/location of the element in the input array.
      ii. If user enters 3, you should take an input of 10 integer elements from the user and store it in an array. Then call **selectionSortDec()** to sort the elements of the array in decreasing order. You should print the array that has been sorted in decreasing order after above function returns. Then you should print the elements of the above array in ascending order as well, without a need to sort it again.
      iii. If user enters 4, the infinite loop must terminate and message should be displayed – "Bye Bye SearchNSort!". And then the program should exit.

Additional Practice Problems

1. For the piece of code shown below that finds out the amount of memory allocated (in bytes) for two arrays, and written as part of `main()`, can you predict what gets printed?

```
int arr[] = {10,20,45,67,68};
printf("Sizeof(arr) = %lu\n", sizeof(arr);
```

Now when the array is passed to a function in which the size is calculated (as shown below), what do you think should get printed?

```
void foo(int arr[], char str[]){
     printf("Sizeof(arr) = %lu\n", sizeof(arr);
}
```

(a) Write a program with these snippets. Call the function appropriately from `main()`, and check whether the outputs you get match what you had predicted.
(b) What can you say about how the name of the array, `arr`, is interpreted by the compiler according to the context of its use?

2. Your task is to design an evaluation system for an online quiz based on multiple choice questions. The program should ask the user to enter the number of questions in the quiz and the correct answer pattern of the questions. 1 (one) mark is given to each correct answer and 0 (zero) for the wrong one. There is no negative marking and 0 marks should be given for un-attempted questions. Each student's answer pattern would be given as input at a time, and evaluated against the "keys" pattern. The output should print total score obtained by the student. After printing the result of the student, the program should ask the user whether he/she wants to continue evaluating another student or stop the program. Make use of arrays to store the keys and the answer pattern of the student entered by the user.

Sample Output:

```
Enter the number of questions (Q): 25
Input key to the quiz:
Keys = abcdabcdabcdabcdabcdabcda
Input the name of the student:  Vivek
Input response of Vivek:
cbcddcbaabcdabcddddddddddd
Marks obtained by Vivek is = 14 out of 25.

Do you want to evaluate another student (Y/N):  Y

Input the name of the student:  Aakash
Input response of Aakash:
cbcddcbaabcdabcddddddddda
Marks obtained by Akash is = 15 out of 25

Do you want to evaluate another student (Y/N):  N
Good Bye!
```

3. Given an unsorted array **a[]** of size **N**, write a program to find its mean and median. The array will be given by the user as an input.

[***HINT***: **Mean** of an array = (sum of all elements) / (number of elements)
The **median** of a sorted array of size **N** is defined as the middle element when N is odd and average of middle two elements when N is even. Since the array is not sorted here, you will have to sort the array first, then apply above formula. ***END OF HINT***]

4. Write a C program that accepts an array **a[]** of size **N** from the user. The program then prints the frequency for each element present in the array.

Sample Test Case: [Text in bold is entered by user]

```
Enter Number of elements: 12
Enter the elements: 2 4 5 6 4 5 5 1 1 2 2 2
2 occurs 4 times
4 occurs 2 times
5 occurs 3 times
1 occurs 2 times
6 occurs 1 times
```

[Note that the order in which you print the elements and their frequency can be different as well. However there must be a separate line for each unique element in the array along with its frequency]