GARRAIO
EMPOWERED BY **INNOVATION**

Lab 3

C Programming

1. Write a function that takes a string (array of char) and returns the string's length.
2. Rewrite Q1 using declaration the string as pointer to char.
3. Write a function that takes two integer pointers as input and swaps the values of the integers they point to.
4. Write a function that reverses the elements of an integer array in-place using pointers.
5. Write a program that reads 10 integers into an array from user and print them in reversing order using pointers.
6. Rewrite Q5 by using two functions, one for reading and another for printing.
7. Write a function that searches for a specific character in a string and returns the number of occurrences using pointers.
8. Write a function that concatenates two strings (null-terminated) and stores the result in a third string using pointers.
9. Write four functions for add, sub, mul and div, then create an array of functions for these implemented functions, let user to choose an index and then go to and perform the specific function.
10. Write a program that calculates the sum of only (even or odd) numbers of an array, using pointer to function. The checking is done on the function and check if it returns true of false.
11. Write a function that swaps two pointers.
12. Write a function that swaps the 2 bytes of a 'short' variable using pointers.

13. Assume that you have a function with the following prototype that sends a byte between two microcontrollers:

    void sendByte(unsigned char Byte);

    Write a function that sends a variable of type 'float' using the above function using two different methods.

14. Assume that a 4-byte unsigned integer value has been transmitted between two microcontrollers of different endianness. Write a function on the receiver side that takes that integer and interprets its original value.

- Group Questions:

15. ..
    a. Write a function that sorts an array of integers ascendingly using any sorting algorithm you know.
    b. Modify the function implemented 15.a to be able to sort both ascendingly and descendingly according to an extra parameter. (Hint: use Enums).
    c. Now write a function that sorts an array of integers ascendingly according to the sum of digits of each integer.
    d. Apparently, there are a lot of different ways to sort an array of integers. Is there a way to make a single function that can sort an array of integers in any way we want? (Hint: use function pointers)

       * Make sure to test your function by comparing its output to the functions implemented in 15.a, 15.b and 15.c.

16. Given the following C struct representing an employee:

```
typedef struct{
    char *name ;
    int age ;
    int SSN ;
    double salary;
 } EmployeeType;
```

a. Write a function that returns a pointer to the first occurrence of an employee with the name "Ahmed" given an array of EmployeeType.

    ```
    * EmployeeType * Find(EmployeeType * employees , int EmployeeCount);
    ```

b. Repeat 16.a but search for the first occurrence of an employee with an age greater than 27.
c. Now modify the above function so that it searches for the first employee that meets both conditions provided in 16.a and 16.b.
d. It looks like we might need to search for an employee according to many different criteria. Can you provide a single function that returns the address of the first employee to meet a given condition or criteria?

    * You can assume that the information provided in an EmployeeType struct is enough to determine if it meets the criteria or not. (Hint: use function pointers).

17. In the following section, we attempt to create a library for performing basic linked list operations. A linked list node can be defined in C as follows:

```
typedef struct Node{
        int data;
        struct Node * next;
}Node;
```

Assume that the linked list provides only a pointer to its head. Implement the following functions:

a. Write a function that inserts a node at the front of the list. Return the new head of the list.

```
Node * pushFront(Node * head , int data);
```

b. Generalize your function in (a) and insert a node at a given position in the list. Return the head of the new list.

```
Node *insertAt(Node *head , int data , int position);
```

c. Write a function that deletes a node at the end of a linked list. Return the head of the new list.

```
Node * popBack(Node *head);
```

d. Generalize your function in (c) and delete a node at a given position in the list. Return the head of the new list.

```
Node* removeAt(Node * head , int position);
```

e. Write a function that prints the elements of a linked list. Use it to validate the functions implemented in (a) to (d). DO NOT USE RECURSION.

```
void print(Node *head);
```

f. Write a function to find the 3rd node from the head of the linked list.

```
Node * getThirdNode(Node * head);
```

g. Write a function to find the nth element from the back of the linked list.

```
Node * getNthNodeFromTheBack(Node *head , int n);
```

h. Write a function that reverses a linked list. Return the head of the new list.

```
Node * reverseLinkedList(Node *head);
```

i. Write a function that detects a cycle in a linked list. It shall return true if there's a cycle, and false otherwise. Remember to include <stdbool.h>.

```
bool hasCycle(Node *head);
```

j. Write a function that returns the middle node of the linked list. Can you do it by traversing the linked list only once ?

```
Node *middleNode(Node *head);
```