Topics: Pointers

1. Using 2D arrays, implement various matrix operations:
    a. Addition
    b. Subtraction
    c. Multiplication
    d. Transpose
    e. Row sum, given a row id
    f. Column sum, given a column id
    g. Diagonal sum (combining major and minor diagonals)
2. Given a square matrix, check if it is an **Identity matrix**. (Search the internet for definition of identity matrix).
3. Implement functions for each of the operations mentioned in **Q.1 and Q.2. Use appropriate parameters.**
4. Implement the functionality of Q.1 and Q.3 now with dynamic allocation. Assume the maximum number of rows is known beforehand, so you can use array of pointers (int* matrix[MAXROW])
5. Now use 2D dynamic memory allocation (int **matrix) for the above tasks. [In this case, you no longer know the maximum number of rows beforehand]
6. Implement the following string related functions using pointer operations. Array indexing is not allowed. Check out Appendix of your book to understand what these functions do. (These are already implemented in the string library of C in the string.h header). But you need to understand the workings and provide your own implementation:
    a. strlen
    b. strcat
    c. strcpy, strncpy
    d. strcmp, strncmp
    e. strrev
    f. strupr
    g. strlwr
    h. strstr
    i. strtok
7. Implement sort function where the comparison logic is encapsulated in a separate function pointed to by a function pointer. Provide 2 different comparison functions for ascending and descending order sorting. Based on input choose the appropriate comparator using the function pointer.

8. Call the qsort library function (stdlib.h) for custom sorting a set of integers.
9. Write code to take a list of names from the console. Use dynamic memory allocation (char **). After taking the input, can you sort the names?
10. Given a string, check whether it is a palindrome or not. Avoid array indexing.
11. Given 2 strings str1, str2, remove all occurrences of str2 from str1. Avoid array indexing.
12. Experiment with passing a pointer to a function
    a. Can you pass a 2d array as an argument to a 2d pointer parameter? Why not?
    b. Can you pass an array of pointers as an argument to a 2d pointer parameter? Why not?