Week 5 - Practice Tasks

Website: www.zypheron.tech **Contact:** info@zypheron.tech

Practice Problem Set: C Programming (Pointers, Memory Management, Double Pointers)

1. **Pointer-Based Value Modification:** Write a C program that declares an integer variable and a pointer to it. Create a function that accepts the pointer and updates the value of the original variable to its square, demonstrating direct memory manipulation via dereferencing.
2. **Swap Function Implementation:** Develop a function named `swap` that accepts the addresses of two integers. The function should exchange the values of these two variables using pointers, ensuring the changes are reflected in the main calling function (Call by Reference).
3. **Array Traversal via Pointers:** Create a C program that initializes an integer array. Use a pointer to traverse the array and print all elements, ensuring that no array indexing syntax (e.g., `arr[i]`) is used, relying solely on pointer arithmetic.
4. **String Length Calculation:** Implement a custom function that calculates the length of a string without using the standard library function `strlen`. The function should accept a `char` pointer and iterate through memory until the null terminator is reached.
5. **Pointer-Driven Maximum Search:** Construct a function that accepts a pointer to an array and its size. The function should traverse the array using pointer arithmetic to identify and return the largest integer present in the dataset.
6. **String Reversal using Two Pointers**: Design a function that reverses a string in place. Utilize the "two-pointer technique" where one pointer starts at the beginning and another at the end of the string, swapping characters and moving inward until they meet.
7. **Dynamic Array Allocation:** Write a program that asks the user for the number of elements they wish to store. Use `malloc` to dynamically allocate memory for an integer array of that specific size, populate it with user input, display the sum, and properly `free` the memory afterward.
8. **Returning Multiple Values:** Create a function named `getGeometry` that accepts the radius of a circle and two float pointers. The function should calculate both the Area and Circumference, storing the results in the addresses pointed to by the arguments, effectively "returning" two values.
9. **Dynamic 2D Matrix Creation:** Develop a C program that utilizes double pointers (`int **`) to allocate memory for a 2D matrix (rows and columns defined by the user). Fill the matrix with sequential numbers, print it, and ensure to free the memory for both the rows and the main pointer to prevent leaks.
10. **Arithmetic Operations via Function Pointers:** Implement a generic calculator program. Create functions for addition and subtraction. In the main function, declare a function pointer, assign it to the appropriate function based on user choice, and execute the operation through the pointer.