Lab Session 7:

Exercise 1: Pointer arithmetic

The program below uses pointer arithmetic to determine the size of a 'char' variable. For a pointer of type char, adding 1 means adding 1 to the address, as char is of 1 byte.

1. Compile and run the program and see what it does.

2. Extend the concept to an integer type pointer to determine the size of an 'int'. Print the value of the pointer before and after adding 1.

3. Apply the same idea to double to figure out how big a double is, by using pointer arithmetic and printing out the value of the pointer before and after adding 1.

4. Explore the effect of adding 2 to the pointers from Step 1 through 3. Discuss this with the student sitting next to you or one of the PGTAs in the lab.

```c
#include <stdio.h>

int main()
{   char ch = 'Z';
    char *cptr = &ch;

    printf("cp is %p\n", cptr);
    printf("The character at cptr is %c\n", *cptr);

    // Pointer arithmetic - check  what cptr + 1 is
    cptr = cptr + 1;
    printf("cptr is %p\n", cptr);

    // Do not print *cptr, because it points to
    // memory space not allocated to your program
    return 0;
}
```

Exercise 2: 1D array with pointer arithmetic

Write a program in C where you declare an array with 10 elements and then print the elements in this array in reverse order using pointer arithmetic.

Exercise 3: 2D array with pointer arithmetic

(In lab session 6, you solved this with array, now we will solve this with pointer arithmetic)

Image is a 2D array and image rotation is a fundamental operation in image processing.

Write a C program that takes an input from user a 3 x 3 2D array, and rotates the **2D array using pointer arithmetic** by

a) 90 degrees clockwise
b) 90 degrees anticlockwise

i. First, write down the pseudo code of your algorithm. Discuss your logic for solving this. Then write the C program to solve it.
ii. Does your code still works if you now take as input a 5 x 5 2D array?

(Optional) Did you solve the above by initialising another 2D array and writing the rotated values to it? If so, try solving it without allocating another 2D array. Modify the input 2D array directly.

Exercise 4: 3D array with pointer arithmetic

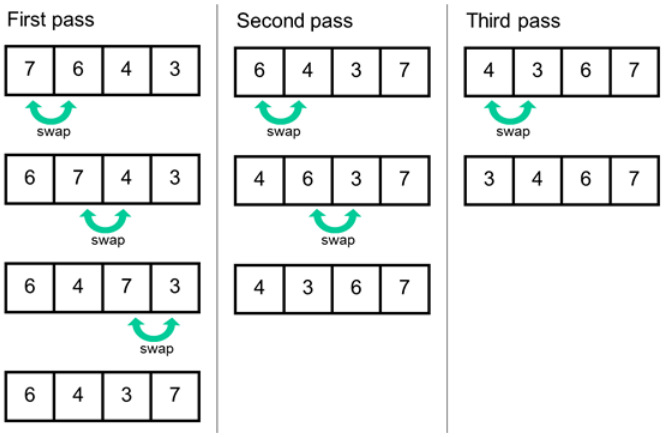
a) Draw a memory representation showing a 3D array (named A) of size 2 x 4 x 4 initialized with ascending numbers starting from 10.. Show how to access the following values using pointer: A[0][0][2]. A[0][2][3], A[1][3][1]

   Discuss this with the student sitting next to you or with one of the PGTAs in the lab.

b) Now write a C program to do the same. And verify your solution.
c) Update your code to print all values stored in A using pointer arithmetic.

Exercise 5: Array sorting using pointers

Given an array of unsorted elements, bubble sort algorithm compares adjacent elements and swaps them if they are in the wrong order.



a) Write the pseudocode for bubble sort.
b) Write a C program that takes the size of the array followed by integer elements in the array as input from the user.

It then implements the bubble sort algorithm as a function to sort this array. Use pointers for array traversal and manipulation.

Exercise 6: (For those who want more!)

Given an array of integers numbers that is sorted in non-decreasing order, find two numbers such that they add up to a specific target number. Let these two numbers be numbers[index1] and numbers[index2] where 1 <= index1 < index2 < numbers.length.

Return the indices of the two numbers, index1 and index2, incremented by one, as an integer array [index1, index2] of length 2.