R4: C Pointers

General Instructions:

1- Create a C project in eclipse called R4
2- Extract the file R4.zip into a folder of your choice at your machine. Copy the files into the R4 project.
3- Rename the file: “R4_template.txt” to “R4.c”. Enter your credentials on top of the file “R4.c”.
4- Follow the videos posted by the instructor on how to complete the tasks presented in this worksheet.
5- To test your code, select the proper function in the file: “R4_test.c”. Compare your results with those presented in “R4_output.txt”. Note that memory addresses are not expected to match.
6- Submit only R4.c file.

Task 1 Learning About Pointers

The concept of pointers is an essential concept in the C language. Since both Java and Python do not have pointers, it is important to spend some time understanding the various ways C pointers are used.

The first two tasks of this worksheet are of greatest important. Make sure that you understand them very well. This will not only save you time in this course, but also welcomes you to the world of good C programmers.

Implement the function `learn_pointers1()` by following the videos posted by the instructor.

This task will demonstrate the following:

1- How to declare and assign a pointer.

2- Difference between different pointer types.
3- How to use the reference operator & for memory addresses
4- How to use the deference operator * to access values stored in memory addresses
5- Understand the memory size of a pointer.

Task 2: Arrays and Pointers

Implement the function `learn_pointers2()` by following the videos posted by the instructor.

This task will demonstrate the following:

1. Understand the relationship between pointers and arrays.
2. Understand pointer arithmetic
3. Learn how to traverse an array using pointers.

Task 3: Pass By Value and By Reference:

The objective of this task is to demonstrate the difference between pass by value and by reference in C language. To achieve that, two swap functions are developed:

1- Pass two integers by value to a swap function (swap1)
2- Pass two integers by references to a swap function (swap2)

Task 4: Reverse An Array

Implement the function `void reverse_array(int* array,const int size)` which reverses a given integer array. The function receives a pointer to an integer array, along with its size. The function prints an error message if the given pointer is NULL or if the given size is non-positive.

Task 5: Swap Pointers:

Implement the function `void swap3(int**,int**)` which swaps two pointers, not their deferenced values. In order to achieve that, the function receives two double pointers.

Task 6: Generic Pointers:

Implement the function `void swap4(void* var1, void* var2, char mode)` which swaps any two values given by the generic pointers. The function works for swapping integers, floats or characters, which is determined by the given mode.

```
/**
 * ------------------------------------------------------
 * Parameters:
 *       var1 (void*)
 *       var2 (void*)
 *       mode (char)
 * returns:
 *       None
 * Description:
 *       Generic swap by reference function.
 *       Supports three modes:
 *       'i': to swap integers
 *       'c': to swap characters
 *       'f': to swap floats
 *       prints an error message for unsupported modes
 * ------------------------------------------------------
 */
```

Task 7: Array of Pointers

Create a function called void `find_max_min`, that takes three parameters:

1- A pointer to an integer array (int* array)

2- The size of the integer array (cons int size)

3- An array of two integer pointers (int* ptr[2])

The function scans through the array and finds the maximum and minimum. A pointer to the maximum value is stored at `ptr[0]` and a pointer to the minimum value is stored at `ptr[1]`.

```
/**
 * -------------------------------------------------------
 * Parameters:
 *          array (int*): an array of integers
 *          size (const int): size of given array
 *          ptr (*int[2]): an array of two pointers
 * returns:
 *          None
 * Description:
 *          Finds the maximum and minimum values of an array
 *          Reference to max value is stored at first element in pointer array
 *          Reference to min value is stored at second element in pointer array
 * -------------------------------------------------------
 */
```