Assignment 3 – Arrays, Structures and Pointers

Academic Honesty

The assignment is individual work. Students are allowed to consult books and online resources but must acknowledge the resources used in the report.

We use MOSS to detect software plagiarism.

Sign the Academic Honesty Pledge in the report.

Problem 1 – Structures

In Assignment 1, problem "Sorted Array", the size of the array (the number of elements) is maintained as an integer separated from the array. When an element is added to or removed from the array, the size is updated in the main() function.

In this assignment, we repeat the "Sorted Array" problem but package the array and its size in a structure named Array. When an integer is added to or removed from the array, the size must be updated in the respective function that adds or removes. Implement the following functions in file arrayStruct.c: **myAdd** ( ), **myRemove** ( ), and **search** ( ).

Assume that all elements in the array are distinct integers sorted in increasing order.

Problem 2 – Arrays and Pointers

In this assignment, we repeat the string functions of Assignment 2, but use dynamically allocated arrays and **only pointers**. Assuming that all input strings are non-empty (having length of at least 1) and shorter than MAX_LENGTH, implement the following string functions in file strgPtr.c:

- **strgLen**(s): return the length of string s.
- **strgCopy**(s, d): copy the content of string s (source) to d (destination).
- **strgChangeCase**(s): for each character in the string, if it is an alphabet, reverse the case of the character (upper to lower, and lower to upper). Keep the non-alphabet characters as is.
- **strgDiff**(s1, s2): compare strings s1 and s2. Return the index where the first difference occurs. Return -1 if the two strings are equal.
- **strgInterleave**(s1, s2, d): copy s1 and s2 to d, interleaving the characters of s1 and s2. If one string is longer than the other, after interleaving, copy the rest of the longer string to d. For example, given s1 = "abc" and s2 = "123", then d = "a1b2c3". If s1 = "abcdef" and s2 = "123", then d = "a1b2c3def".

Important: **Do not use array indexing in file strgPtr.c.** Make sure that your submitted file does not contain the square brackets, even in the comments. The grading script will use the "grep" command to search for the square brackets and give zero to a program when it finds a square bracket.

Common Notes

- Do not use any C library function at all in your code. Do not add any header file to the code.
- The functions (algorithms) must be the most efficient in terms of running time and memory usage.
- Submit files arrayStruct.c and strgPtr.c. Complete the header with your student and contact information. Include sufficient comments in your code to facilitate code inspection.
- **Submit a report in PDF** with following information: references (sources); error conditions and actions taken; brief algorithm; running time of the function (algorithm) and a brief explanation. See the template a3report.docx for an example.
- See files a1output.txt and a2output.txt posted earlier for sample input and output.
- Your code will be graded automatically by a script, so make sure to strictly follow the specifications.
- Do not use any output statements (for example, printf) in your code, or they will mess up the grading scripts.
- Use the main( ) functions provided to test your code. Understanding the code in the main( ) functions is part of the assignment. Do not change the code in the main( ) functions in the final submission, or your program will mess up the grading script.
- C compilers are machine-dependent and programs may behave differently on different systems. C assignments will be graded on an EECS server from the command line by a script. To make sure your code pass these tests, it should be tested on an EECS server (for example, red.eecs.yorku.ca) before the final submission.
- We will use more test cases for grading.