CSE102 – Computer Programming

Homework #11

Due Date: **02/06/2020**

Hand in: A student with number 20180000001 should hand in a file named 20180000001.c for this homework.

Write a complete program accomplishing all the tasks below.

Part 1 (65 pts) The program reads a text file named 'list.txt'. This file contains a great number of comma seperated random positive integers (e.g., N=1000000). The program should read this file only *once* and store these numbers in the given order. Reading the file once means that you cannot read any portion of the file twice. This means that during reading you cannot use large fixed-sized arrays. You will do this reading in two different ways, hence two separate functions.

- `… read_array(char *filename, …)`: This function will do the task described above. It will take a filen name and return a dynamicaly allocated integer array including the read numbers. You can indicate the end of the array using the value -1. Your total memory (stack or heap) usage in this function should not be more than 4*(N+1) + 1000 bytes (assuming 4-byte integers are used). You are expected to write the fastest function possible.

  You are expected to properly define the input arguments and return values for this function.

  Hint: Consider the method discussed during the class.

- `… read_linkedlist(char *filename, …)`: Similar to the first function above, this function will read the given file and return the the sequence of numbers read from the file in a linked list. Each node of the list should only hold 1 integer and a pointer. Your total memory (stack or heap) usage in this function should not be more than (4+4)*N + 1000 bytes (assuming 4-byte integers and addresses are used). You are expected to write the fastest function possible.

  You are expected to properly define the input arguments and return values for this function.

Your program will call these two functions once for each. The returned array and linked list will be sent to two other functions:

- `float * stats_array(…)`: Given the numbers in an array, this function returns an array of 4 numbers. These are mean, max, mean and std deviation of these numbers. Make sure that you properly allocate the return array in the function.
- `float * stats_linkedlist(…)`: Given the numbers in a linked list, this function returns an array of 4 numbers. These are mean, max, mean and std deviation of these numbers. Make sure that you properly allocate the return array in the function.

Your main function will:

- Call the read_array and read_linkedlist functions each once.
- Measure the time taken for these functions. Here, you can use start() and end() functions (returntype: clock_t) from the <time.h> library. These functions will return the clock cycles that the program needed to run the tasks. You can use CLOCKS_PER_SEC definiton to convert it into seconds. The screen output should have the type of seconds.

- Print the time required for each of these functions.
- Print the total memory in bytes required for each of the dynamic array and linked list.
- Call the stats_array and stats_linkedlist functions.
- Measure the time taken for these two functions.
- Print the time required for each of these functions.
- Report the results of these two comparisons.

Note that operating systems may cache during the file reading. Therefore, when you test the speed of these functions, the order in which you call them may give differing results. To prevent biasing one method over the other in speed, repeat the above reading steps a few times in different orders. For example make the following calls: read_array, read_linked_list, read_linked_list, read_array, read_array, read_linked_list. And, average the timing results of the last four calls.

Other rules for this part:

- ❖ You can assume that the length of the text of the file is 1.000.000.
- ❖ You are not allowed to use any library other than stdio.h, string.h, stdlib.h, and time.h.
- ❖ You can use the 'strtol' function from <stdlib.h> library to parse a string to int.
- ❖ You can write your own functions to make things easier.

Part 2 (35 pts) Write a function that takes two arguments holding a large number of integers. The first argument is an array allocated from the heap, and the second is a linked list (use the same declarations and assumptions for these arrays as in Part 1). It is assumed that these two arguments are holding the same sequence of integers. However, for some reason, a few of the entries are entered wrong. For example:

Dynamic array : {11, 2, 35, 14, 17, 67, 55, 98, 32, 22}

Linked list : {11, 2, 35, 13, 17, 67, 55, 99, 32, 22}

Your function will find the difference in the entries and return them in a separate array of "struct { int n1, n2; }". This array is expected to be created dynamically (no fixed array declerations).

You are asked to figure out a way both to generate these arrays and to change some of the values dynamically and finally print out the array that has been returned from the function.

The program should perform Part 1 and Part 2 respectively.

General Rules:

1. The program must be developed on Linux based OS and must be compiled with GCC compiler, any problem which rises due to using another OS or compiler won't be tolerated.
2. Note that if any part of your program is not working as expected, then you can get zero from the related part, even it's working in some way.
3. Upload your .c file on to Moodle to deliver your homework. Name format can be found on the top of this homework sheet.
4. You can ask any question about the homework by sending an email to sgulmez2018@gtu.edu.tr or by using the forum in the Moodle page of the course.