CIS 26B
Advanced C
Programming Assignments

Homework 1

100 Points

Arrays, Strings, Structures, Sorting, Pointers, and Dynamic Allocation of Memory

Projects:

A. 26B_Hw_1_A.c – see the beginning comment

B. 26B_Hw_1_B.c Create and process arrays of structures – see program requirements on the next page.

Grading Hw_1_A	**(25Points)**
1. Source code comments	– 5 Points
2. The printList() function	– 5
3. Second sort	– 10
4. Updated main()	– 5

Grading Hw_1_B	**(75Points)**
1. Source code comments	– 5 Points
2. Read file names	– 5
3. Read from file	– 20
4. Sort	– 15
5. Display array	– 10
6. Write to file	– 15
7. No memory leak	– 5 // – see details on the next pages

NOTE: Write a comment in the beginning of the program. Write a comment for each function. Write comments inside functions (if needed). Use proper indentation and spacing. Do not use global variables. Do not use the goto statement. Always check if opening an input file was a successful operation. Do the same for dynamic allocation of memory.

B. 26B_Hw_1_B.c Create and process arrays of structures

Write a program which expects the name of an input file and an output file to be given by the user. If the user does not input any names, default file names should be used, such as **`in.txt`**, and **`out.txt`**. The input files have lines which look like this:

```
MSFT 150
```

The string represents a stock symbol. The number represents the number of shares bought for that stock.

Any stock/shares can be represented any number of times in the input file. Your program will create an output file which contains each stock name once followed by the number of shares. Here is an example. If the input file contains the following data:

```
MSFT 150
AAL 280
MSFT 100
AAL 30
MSFT 200
```

the output file will contain each stock symbol once followed by the total number of shares for that stock:

```
AAL 310
MSFT 450
```

Read data from the first input file into a dynamically allocated array of STOCK structures. You may assume that the maximum size of a name string is 25. The program should use either the insertion sort algorithm or the selection sort algorithm to sort the array in ascending order by stock name. To demonstrate that the sorting algorithm works, display the sorted array to the screen. Create your own input file using the data shown on the next page. On the first line in the input file provide the number of stock/shares lines. Make sure that your program does not produce memory leaks. Memory leak detection is optional (see last page). Run the program once and save the output at the end of the source file as a comment. Compress the source file, input and output files and upload the compressed file: 26B_LastName_FirstName_H1.zip

in.txt

```
44
MSFT 150
AAL 199
CHTR 280
YHOO 89
XLNX 27
EBAY 899
MSFT 130
BBBY 80
TSLA 45
SIRI 25
CHTR 143
NLFX 133
AAPL 445
SIRI 15
YHOO 78
NLFX 89
XLNX18
AAPL 2345
TSLA 234
SIRI 34
YHOO 45
SIRI 523
NLFX 1234
TSLA 50
GOOG 123
YHOO 99
BBBY 211
TSLA 67
XLNX 122
NLFX 18
TSLA 452
XLNX 83
TSLA 125
YHOO 147
MSFT 100
XLNX 10
AAL 170
TSLA 67
GOOGL 123
XLNX 90
SIRI 1311
TSLA 343
AAPL 89
```

BBBY 180

Memory Leak Detection

It is a good habit to release the memory when it is no longer needed.

"Memory leaks are among the most difficult bugs to detect because they don't cause any outward problems until you've run out of memory and your call to **`malloc`** suddenly fails. In fact, when working with a language like C or C++ that doesn't have garbage collection, almost half your time might be spent handling correctly freeing memory. And even one mistake can be costly if your program runs for long enough and follows that branch of code."

Windows, Microsoft Visual Studio:
To check if memory was released properly, use CrtDumpMemoryLeaks as described below:

```
//...
  printf( _CrtDumpMemoryLeaks() ? "Memory Leak\n": "No
Memory Leak\n");
     return 0;
} // end of main()
```

`_CrtDumpMemoryLeaks` is a debug function:
. returns TRUE if a memory leak is found;
. otherwise, the function returns FALSE.

Required Header: `#include <crtdbg.h>`

Read more about finding memory leaks with the CRT library:
https://docs.microsoft.com/en-us/visualstudio/debugger/finding-memory-leaks-using-the-crt-library?view=vs-2017

Unix
VALGRIND (free download: http://valgrind.org)
Tutorial:
https://www.cprogramming.com/debugging/valgrind.html