Coding Assignment 5

For Coding Assignment 5, you are writing the code from the final question of Exam 2 and adding some features.

High Level Description

Your program will read in a file of lunch orders. It will store them in an array. The orders will be displayed to the user and the user will be asked to supersize as many orders as they want. Once the supersizing process is completed, the user will be asked to sort the orders using different criteria before writing them back out to a new file.

Objectives

The objectives of this assignment are to learn to use the following coding constructs

function pointers

void pointers

qsort()

command line parameters

file handling – opening, closing, reading and writing

strtok()

`makefile`

Copy your `makefile` from the previous assignment and modify as needed. Your `makefile` should compile/link `Code5_xxxxxxxxxx.c` and `SortLib.c`. `SortLib.h` should be in the `makefile`.

`SortLib.h`

Create a file named `SortLib.h`. Add an include guard. Put the typedef for `Combos` in this file. Put the prototypes for the `qsort()` compare functions in this file.

`SortLib.c`

Create a file named `SortLib.c`. Include `SortLib.h`. Put the code for the qsort() compare functions in this file.

Command Line Parameters

Create a function to read the command line parameters and store the values. Get the name of the input file and of the output file. If you are not using variable command line parameters (see Bonus below), then the order in which you read the command line parameters should be input file and output file. Your code will be graded with this assumption.

Bonus (10 points) – use variable command line parameters as show in the output. If no parameters are passed, your program should print a message stating what the expected parameters are. Your code will be graded assuming your command line parameters can be run in any order.

`ReadLunchOrders`

Create a function called `ReadLunchOrders`. It should take in the array of lunch orders and the pointer to the file that contains the lunch orders. Function should return the number of orders read from the file. The function should read all records in the file. `strtok()` must be used to parse each line of the file and separate each line into members of the

structure. The information in each line of the file will be separate by pipe symbols. `ferror()` must be used to check for file read errors. `perror()` should be used to print an error message. If an error occurs, return -1 for the number of orders.

PrintLunchOrders

Create a function called `PrintLunchOrders`. It should take in the array of lunch orders and the number of orders in the file. Print out the lunch orders. Format the output as shown in the examples.

WriteLunchOrders

Create a function called `WriteLunchOrders`. It should take in the array of lunch orders, the number of orders in the file and the pointer to the file to which the new orders will be written. It does not return anything. Each line should be written to the file using the same pipe delimited format as the original file. `ferror()` must be used to check for file write errors. `perror()` should be used to print error messages.

SuperSizeIt

Create a function called `SuperSizeIt`. It should take in a single lunch order (pointer). It does not return anything. Update `fry_size` to 'L' and `drink_size` to 'L'. Use pointer notation to access the structure members.

main

Create an array of size 1000 to hold the lunch orders read in from the file.

Create an array of function pointers and initialize it with the compare functions.

Call the function to read in the command line parameters.

Open the input order file with a mode of read only. If it does not open, print a message and exit.

Open the output order file with a mode of write only. If it does not open, print a message and exit.

Call function `ReadLunchOrder` and store the return value

Close the input order file.

Create a loop to

- call `PrintLunchOrders`
- print the Super Size prompts and prompt for which combo to super size
- Ensure that the chosen menu option is valid – user should reenter until choice is valid
- Call function `SuperSizeIt` and pass the order that needs to be supersized (a single element of the array)
- The user should be allowed to update as many as orders as wanted

Print the sort choice menu

Ensure that the chosen menu option is valid – user should reenter until choice is valid

If user chooses to sort, then call `qsort()` with using an element of the array of function pointers. No matter which sort is picked, only one call to `qsort()` is allowed in the program.

Call function `WriteLunchOrders`

Close the output file

Testing

Run your `Code5.e` and confirm that your output matches the output in the assignment. Confirm that you have met all elements of the rubric.

The GTAs will grade your code using the rubric. One of the rubric steps will be

1. They will run your program with a much larger lunch order file. The size of the individiual fields will not exceed those show in the sample output.
2. They will supersize several orders, sort the orders and write them to your output file. They will then run your program again and use the output file as the input file as shown in the same output.
3. Your program should be able to use the prior run's output file as an input file and should display the records in the order in which they were sorted into during the previous run.

Code Submission

Submit a zip fle containing the following files

`Code5_xxxxxxxxxx.c`

`SortLib.c`

`SortLib.h`

`makefile`

CODING

20 points

Write a complete C program to do the following

1. Create a structure named `Combos` with 5 fields – name, sandwich, fry size, drink size, drink type. See sample output below to determine the types and sizes of the fields.
2. In `main()`, declare and initialize an array called `LunchOrders` of size 3 of type `struct Combos`. This must be done with one statement.

Name	Sandwich	Fry Size	Drink Size	Drink Type
Ronald	Big Mac	L	L	Coke
Wendy	Cheeseburger	S	S	Diet Coke
Jack	Filet-O-Fish	M	M	Sprite

3. In `main()`, call function `PrintLunchOrder` to print the whole lunch order (pass and print entire array).
4. In `main()`, ask which order to supersize.
5. In `main()`, call function `SuperSizeIt` to set fry size and drink size to L for the order to supersize (pass and update a single element of the array).
6. In `main()`, call function `PrintLunchOrder` again to show `SuperSizeIt` changes.

Sample Output

```
Ronald  Big Mac         L       L       Coke
Wendy   Cheeseburger    S       S       Diet Coke
Jack    Filet-O-Fish    M       M       Sprite
Supersize which order? (1-3) 2
Ronald  Big Mac         L       L       Coke
Wendy   Cheeseburger    L       L       Diet Coke
Jack    Filet-O-Fish    M       M       Sprite
```

```
[frenchdm@omega CA5]$ more LunchOrder.txt
Ronald|Big Mac|L|L|Coke
Wendy|Cheeseburger|S|S|Diet Coke
Jack|Filet-O-Fish|M|M|Sprite
[frenchdm@omega CA5]$ Code5_1000074079.e INITIALORDERS=LunchOrder.txt FINALORDERS=FinalOrders.txt
                                        Fry     Drink   Drink
        Name            Sandwich        Size    Size    Type
1.      Ronald          Big Mac         L       L       Coke
2.      Wendy           Cheeseburger    S       S       Diet Coke
3.      Jack            Filet-O-Fish    M       M       Sprite

Enter 0 to finalize orders and print final orders to file

Supersize which order? (1-3) 2
                                        Fry     Drink   Drink
        Name            Sandwich        Size    Size    Type
1.      Ronald          Big Mac         L       L       Coke
2.      Wendy           Cheeseburger    L       L       Diet Coke
3.      Jack            Filet-O-Fish    M       M       Sprite

Enter 0 to finalize orders and print final orders to file

Supersize which order? (1-3) 0

Choose a sort before writing out the file

0. No sort - write out in current order
1. Sort by Name
2. Sort by Sandwich
3. Sort by Fry Size
4. Sort by Drink Size
5. Sort by Drink Type

Enter choice 1


Writing lunch orders to file....

[frenchdm@omega CA5]$ more FinalOrders.txt
Jack|Filet-O-Fish|M|M|Sprite
Ronald|Big Mac|L|L|Coke
Wendy|Cheeseburger|L|L|Diet Coke
```

Bonus – running program with no parameters should print instructions to user

```
[frenchdm@omega CA5]$ Code5_1000074079.e
Run command is

Code5_1000074079.e INITIALORDERS=file1 FINALORDERS=file2
```

Run command with bonus

```
Code5_1000074079.e INITIALORDERS=LunchOrder.txt FINALORDERS=FinalOrders.txt
```

or

```
Code5_1000074079.e FINALORDERS=FinalOrders.txt INITIALORDERS=LunchOrder.txt
```

Run command without bonus

```
Code5_1000074079.e LunchOrder.txt FinalOrders.txt
```