CS 2413 - 501/503 - Project 2

This is the second programming assignment of the semester, it is designed to test your programming abilities before the second exam. It will heavily involve pointers, strings, trees, structs, and file I/O.
Your assignment must be submitted as a .c file, any other format will result in a penalty to your grade, the assignment is due by 10/26 at midnight.

Project Specification

You are to implement a system that stores customer loyalty accounts at a grocery store, then processes basic orders for said customers.

Customers' loyalty accounts will have the following:

- ***ID***
    - A unique 4 digit ID number (0-9999). Do not worry about trailing zeros.
- ***Name***
    - The customer's name (these do not have to be unique).
- ***Store Balance***
    - Amount of store credit they have accrued.

The customers' accounts should be stored in a BST, use the customer's ID as the key to compare against when creating this tree. The system should allow for 4 operations to happen:

1. Add a customer to the system
    a. Complete with name, ***ID***, and balance
2. Remove a customer from the system
    a. An ***ID*** will be given to do this
3. Process a list of orders for customers in our system
    a. A text file (to be given via the console) will contain a series of orders for customers consisting of ***ID***s and cash totals for the order
    b. Go through the orders one by one, subtract the order total from the customer's ***Store Balance*** first, then take payment for the remainder.
    c. After each order, print a message to the console detailing the order
4. Close down the system
    a. Prompt the user to select, *pre-order, in-order,* or *post-order.*
    b. Write the current BST to a file "`newcustomers.txt`" in the order specified in the step above.
    c. Close the program

The syntax for these commands, and the details for the order message will be given below.

Your program will read in customers from a file "`customers.txt`", parse them as structs then place them into your BST with the ***ID*** of each customer as the key.

The syntax for the operations listed above are as follows:

1. `add` *num name balance*
    a. int - *num* is a valid, unique ***ID*** number as detailed on page 1.
    b. string - *name* is the ***name*** of the customer
    c. float - *balance* is the amount (in $) of ***Store Balance*** the customer will have
2. `delete` *num*
    a. int - *num* is the ***ID*** number of an existing customer as detailed on page 1.
3. `process` *filename*
    a. string - *filename* is the name of the file which contains the orders as detailed in step 3 of page 1.
4. `quit`
    a. This will prompt a message "`Please enter output format: pre, post, or in order`"
    b. User should enter "`pre`", "`post`", or "`in`" to indicate the order they want.
    c. Please add trailing zeros to the customer's ***ID*** when printing to "`newcustomers.txt`". I.e. 12 -> 0012

The console message for each order in the process routine should look as follows:
"Customer *NUM*, *NAME*, payed $*BAL* from their store balance and $*CASH* at the register, $REM remaining store balance"

- *NUM* is the customer's ***ID***
- *NAME* is the customer's ***name***
- *BAL* is the amount deducted from the customer's ***Store Balance***. The price of the item should always try to deduct from the ***Store Balance*** before taking money at the register.
- *CASH* is the amount the customer has to pay after exhausting their ***Store Balance***.
- *REM* is the amount left in the customer's ***Store Balance*** after the transaction

Note: *BAL*, *CASH*, or *REM* can each be 0.00.

This message should be printed to the console for *each* order processed in the file.

You must use a ***binary search tree*** as defined in the book/slides to use for the customers' queue. The key for the BST should be the customer's ***ID***.

You must use a **struct** as defined in the book/slides for the customers in the BST. You can add whichever data you want to this to help with traversal.

Example input/output:

This will be in a file "customers.txt" that will be in the same directory as your .c file.

```
{1688, "Karen", 65.00}
{0112, "Bob", 15.00}
{5555, "T.J. Green", 0.00}
{9000, "Karen", 10.00}
{0001, "Human McPerson", 150.00}
```

This would indicate:

Karen has ***ID*** 1688, and ***Store Balance*** $65.00
Bob has ***ID*** 0112, and ***Store Balance*** $15.00
T.J Green has ***ID*** 5555, and ***Store Balance*** $0.00
Karen has ***ID*** 9000, and ***Store Balance*** $10.00
Human McPerson has ***ID*** 0001, and ***Store Balance*** $150.00

Sample Add:

```
input> add 0002 Billy 61.00
```

This should add a customer to the BST with ***name*** "Billy", ***ID*** 0002, and ***Store Balance*** $61.00.

```
input> add 0001 Billy 61.00
```

This should print an error, as ***ID*** 0001 is already taken.

Sample Delete:

```
input> delete 0001
```

This should delete Human McPerson's account from the BST (make sure to reform the tree appropriately)

This is a sample file with customer orders "orders.txt". Remember this filename should be input at the console and can be named anything other than "customers.txt" or "newcustomers.txt":

```
{1688, 88.88}
{0112, 14.99}
{5555, 10.50}
{0001, 99.99}
```

This would indicate that:

Karen (1688) is buying $88.88 worth of items.
Bob (0112) is buying $15.00 worth of items.
T.J. Green(5555) is buying $10.50 worth of items.
Human McPerson(0001) is buying $99.99 worth of items.

The log output for these three people would be:

Customer *1688*, *Karen*, payed $*65.00* from their store balance and $*23.88* at the register, $0.00 remaining store balance
Customer *0122*, *Bob*, payed $*14.99* from their store balance and $*0.01* at the register, $0.00 remaining store balance
Customer *5555*, *T.J. Green*, payed $*0.00* from their store balance and $*10.50* at the register, $0.00 remaining store balance
Customer *0001*, *Human McPerson*, payed $*99.99* from their store balance and $*0.00* at the register, $50.01 remaining store balance

Sample Shut-Down:

```
input> quit
output> Please enter output format: pre, post, or in order
input> post
output> writing to newcustomers.txt in post-order, shutting down...
```

Note that when the program is closed down, the *updated* customer data should be written to "`newcustomers.txt`" in whichever order is specified at shut-down.

Submission Instructions

Upload your program as *[eraider]_storesystem.c* where [eraider] is your eraider id. Upload your program to Blackboard where you found this specification.

Your program should run without any user input, as all input will come from `"customers.txt"`. I will open your .c file, make sure the correct input file is there for me to test, and run your program. You may include any functions or print statements as you wish to help you. Make sure you output the log as defined above to `"newcustomers.txt"` in the same directory as the .c file.