/\*

\* Name: Ebony Cross

\* Class: CS 226

\* Project 10B: chapter 13

\* Instructor: Tsai

\* Description: Implement user input, file input/output streams, validation, while loops, functions, pointers, implementation of classes, structures, arrays, searching/sorting and displaying module menus

\* Due: 11/23/2015

\* Platform/Compiler: Xcode

\*/

//delcare directives

#include <iostream>

#include "mainmenu.h"

#include <iomanip>

#include <string>

#include <cstring>

#include <ctime>

#include <cctype>

#include <fstream>

#include <sstream>

#include <cstdlib>

using namespace std;

//global variables

const int SIZE = 20;

int subscript; //subscript number to store elements in corresponding arrays

const int SPACES = 7; //constant for formatting menu

BookData list[SIZE];

fstream dataFile;

bool found;

BookData temp;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*The default constructor that set members to null/zero \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BookData::BookData()

{

\*(dateAdded) = NULL;

\*(publisher) = NULL;

\*(author)= NULL;

\*(isbn) = NULL;

\*(bookTitle) = NULL;

qtyOnHand = 0;

wholesale = 0.0;

retail = 0.0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*The constructor that will acceept the members of the class\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

BookData::BookData(char t[], char i[], char a[], char p[], char d[], int q, double w, double r)

{

strcpy(bookTitle, t);

strcpy(isbn, i);

strcpy(author, a);

strcpy(publisher, p);

strcpy(dateAdded, d);

qtyOnHand = q;

wholesale = w;

retail = r;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*case converters\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*convert each letter of sentence to uppercase letter\*/

void strUpper(char \*ptr)

{

while (!(\*ptr == '\0')){

\*ptr = toupper(\*ptr);

ptr++;

}

}

/\*convert each letter of sentence to uppercase letter\*/

string convert(string s) {

for (int index =0; index < s.length(); index++){

s[index] = tolower(s[index]); //convert to lower case, one letter at a time

}

return s;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*set book data\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void BookData::removeBook(BookData s[], int index)

{

\*(s[index].bookTitle) = NULL;

}

/\*set book title\*/

void BookData::setTitle(char a[])

{

strUpper(a);

strcpy(bookTitle, a);

}

/\*set isbn number\*/

void BookData::setISBN(char a[])

{

//prompt user for isbn

strUpper(a);

strcpy(isbn, a);

}

/\*set author name\*/

void BookData::setAuthor(char a[])

{

strUpper(a);

strcpy(author, a);

}

/\*set publisher's name\*/

void BookData::setPub(char a[])

{

strUpper(a);

strcpy(publisher, a);

}

/\*\*set date added\*/

void BookData::setDateAdded(char a[])

{

strcpy(dateAdded, a);

//strUpper(a);

}

/\*set quantity on hand\*/

void BookData::setQty(int s)

{

qtyOnHand = s;

}

/\*set wholesale value of book\*/

void BookData::setWholesale(double s)

{

wholesale = s;

}

/\*set retail value price of the book\*/

void BookData::setRetail(double s)

{

retail = s;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*isEmpty\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int BookData::isEmpty(BookData list[], int size){

int count = -1;

for (int index = 0; index < size;) {

if(!(strlen(list[index].getTitle()) == 0))

{

// cout << "full" <<endl;

index++;

}

else if ((!(strlen(list[index].getTitle()) == 0) && index == 19))

{

cout << "The inventory is full. No more books may be added to the inventory" << endl;

count = -1;

return count;

}

else{

//cout << "count is: " << count << endl;

count = index;

return count;

}

}

return count;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Read & Write File\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*-------------------openInFile-----------------------\*/

bool openFileIn(fstream &file, string fileName)

{

file.open(fileName.c\_str(), ios::in | ios::out);

if (file.fail())

{

return false;

}

else

{

return true;

}

}

/\*-------------------writeToFile-----------------------\*/

void writeToFile(fstream &file, string fileName)

{

char arrEnd[] = {'\0'};

file.open(fileName.c\_str(), ios::out);

cout << "writing to " << fileName << "..." << endl;

file << "ISBN,Title,author,Publisher,Date added,Quantity-on-hand,Wholesale Cost,Retail price" << endl;

for (int index = 0; index < SIZE; index++)

{

//char title = list[index].getTitle();

if (strcmp(list[index].getTitle(), arrEnd) != 0) { //maybe try calling isEmpty

//display book information

file << list[index].getIBSN() << ",";

file << list[index].getTitle() << ",";

file << list[index].getAuthor() << ",";

file << list[index].getPub() << ",";

file << list[index].getDateAdded() << ",";

file << list[index].getQty() << ",";

file << list[index].getWholesale() <<",";

file << list[index].getRetail()<< endl;

}

}//end of for-loop

cout << "Done.\n";

file.close();

}

/\*-------------------readInFile-----------------------\*/

void addFileContents(fstream &file){

BookData book;

string line, word;

stringstream x;

int lineSize = 8;

int index = 0;

int subscript;

int count;

if(file){

getline(file,line);

//while the last read operation was successful, contine

while (getline(file,line)){

x << line;

cout << "\nline " << index << ": " << line << "\n\n";

for(int i = 0; i < lineSize; i++)

{

getline(x, word, ',');

if ( i == 0){

char isb[14];

strcpy(isb,word.c\_str());

book.setISBN(isb);

}

else if ( i == 1){

char t[51];

strcpy(t,word.c\_str());

book.setTitle(t);

//cout << "Title for the book is: " << book.getTitle() << endl;

}

else if ( i == 2){

char a[31];

strcpy(a,word.c\_str());

book.setAuthor(a);

}

else if ( i == 3){

char p[31];

strcpy(p,word.c\_str());

book.setPub(p);

}

else if ( i == 4){

char d[11];

strcpy(d,word.c\_str());

book.setDateAdded(d);

}

else if ( i == 5){

int qty;

qty = atoi(word.c\_str());

book.setQty(qty);

}

else if (i == 6){

double cost;

cost = atof(word.c\_str());

book.setWholesale(cost);

}

else if ( i == 7){

double price;

price = atof(word.c\_str());

book.setRetail(price);

}

cout << "word " << i << ": " << word << endl;

}

char isb[14];

strcpy(isb, book.getIBSN());

count = searchISBN(list, SIZE, isb);

if (count == -1) {

subscript = book.isEmpty(list, SIZE);

list[subscript] = book;

cout << "\n\ntitle at index " << subscript << ": " << list[subscript].getTitle() << endl;

}

else{

cout << "match found in inventory.";

}

x.clear();

index++;

}

//close file

file.close();

}

else

{

cout << "Error.\n";

}

}

/\*-----------------------Search file--------------------------------\*/

int searchFile(fstream &file, string name, char value[], int size){

BookData book;

string line, word;

stringstream x;

int lineSize = 8;

int index = 0;

int position = -1;

int counter = 0;

int choice;

char title[51];

char \*strPtr = NULL;

strUpper(value);

cout << "looking up: " << value << endl;

cout << "Searching file for a match..." << endl;

cout << "opening file...\n";

file.open(name.c\_str(), ios::in);

if(file){

getline(file,line);

//while the last read operation was successful, contine

while (getline(file,line)){

x << line;

cout << "\nline " << index << ": " << line << "\n\n";

for(int i = 0; i < lineSize; i++)

{

getline(x, word, ',');

if ( i == 1){

strcpy(title,word.c\_str());

book.setTitle(title);

cout << "Title in store is: " << book.getTitle() << endl;

cout << book.getTitle();

strPtr = strstr(title, value);

if (strPtr != NULL){

cout << "\nBook title: " << book.getTitle() << " at index " << counter << endl;

cout << "Is this the book you were searching for?" << endl;

cout << "Enter 1 for YES | Enter 9 for NO\n";

cout << "Response: ";

cin >> choice;

cout << endl;

if (choice == 1){

position = counter;

file.close();

return position;

}

else if (choice == 9)

{

cout << "inventory search will resume...\n";

}

}

}

}

x.clear();

index++;

counter++;

}

//close file

file.close();

cout << "\nDone. Closing file...\n";

}

else{

cout << "Error.\n";

}

return position;

}

/\*-----------------------Search structure--------------------------------\*/

int searchISBN(BookData list[], int size, char value[]){

int position = -1;

int index = 0;

strUpper(value);

cout << "looking up: " << value << endl;

cout << "Searching for match in inventory...";

for(index = 0; index < size; index++){

//if a match is found, do the if statement

if (strcmp(list[index].getIBSN(), value) == 0){

cout << "\nISBN #: " << list[index].getIBSN() << " at index " << index << endl;

cout << endl;

position = index;

return position;

}

}

cout << "match not found." << endl;

return position;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TotalSumRet\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*uses a pointer to accept the address of an array.

Then returns the total of the elements in the array.\*/

double totalSumRet(BookData \*arr, BookData \*arr\_ptr2, int size)

{

double sum = 0.0;

for (int count = 0; count < SIZE; count++){

sum += (\*arr).getRetail() \* (\*arr\_ptr2).getQty();

arr++;

arr\_ptr2++;

}

return sum;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*TotalSum\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*uses a pointer to accept the address of an array.

Then returns the total of the elements in the array.\*/

double totalSum(BookData \*arr, BookData \*arr\_ptr2, int size)

{

double sum = 0.0;

for (int count = 0; count < SIZE; count++){

sum += (\*arr).getWholesale() \* (\*arr\_ptr2).getQty();

arr++;

arr\_ptr2++;

}

return sum;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*search book title\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int searchList(BookData title[], int size, char value[]){

int index = 0; //used as a subscript to search array

int position = -1; //to record position of search vaule

bool found = false; //flag to indicate if the value was found

//cout << "book title: " << bookTitle[0] << endl;

strUpper(value);

// cout << "value as lowercase: " << value << endl;

while (index < size && !found){

//string newTitle = convert(list[index]);

// cout << "new title: " << newTitle << endl;

if(strcmp(title[index].getTitle(),value)== 0){

found = true;

cout << "Book has been found in the inventory at index " << index << endl;

position = index;

return position;

}

cout << "search list:" << title[index].getTitle() << endl;;

index++;

}

return position;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*main menu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*main menu function that calls functions of: reports, invmenu, bookinfo and cashier while looping through display menu\*/

int main() {

int choice; //allows user to enter choice

//display menu options repeatedly until user selects item #4

do

{

choice = temp.printMainMenu();

//implement main menu selection

switch (choice) {

case 1:

cashier(); //call function

break;

case 2:

invmenu(); //call function

break;

case 3:

reports(); //call function

break;

default:

break;

}

} while (choice != 4);

return 0; //end main

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Reports\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Implement user input for report function via validation, while loops and display module menus\*/

void reports() {

int choice; //user's input

do

{

choice = temp.printReportMenu();

//display the user's menu selection

switch (choice){

case 1: repListing(); //calling function

break;

case 2: repWholesale(); //calling function

break;

case 3: repRetail(); //calling function

break;

case 4: repQty(); //calling function

break;

case 5: repCost(); //calling function

break;

case 6: repAge(); //calling function

break;

case 7: cout << setw(SPACES -1) << "" << "You selected item " <<choice << ". Exiting the program\n";

break;

default: cout << setw(SPACES -1) << "" << "That is an invalid choice.\n";

}

cout << "" << endl;

} while (choice != 7);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*repListing\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*function #1 for reports menu\*/

void repListing(){

int response;

// current date/time based on current system

time\_t now = time(0);

char arrEnd[] = {'\0'};

// convert now to string form

char \* dt = ctime(&now);

cout << "You selected Inventory Listing." << endl;

cout << "Here is the Inventory Listing Report on " << dt << endl;

cout << "--------------------------------------------------------\n";

for (int index = 0; index < SIZE; index++)

{

if (strcmp(list[index].getTitle(), arrEnd) != 0) {

//display book information

cout << setw(2) << "" << "ISBN: " << list[index].getIBSN() << endl;

cout << setw(2) << "" << "Title: " << list[index].getTitle() << endl;

cout << setw(2) << "" << "Author: " << list[index].getAuthor() << endl;

cout << setw(2) << "" << "Publisher: " << list[index].getPub() << endl;

cout << setw(2) << "" << "Date Added: " << list[index].getDateAdded() << endl;

cout << setw(2) << "" << "Quantity-On-Hand: " << list[index].getQty() << endl;

cout << setw(2) << "" << "Wholesale Cost: " << list[index].getWholesale() <<endl;

cout << setw(2) << "" << "Retail Price:" << list[index].getRetail() << endl;

cout << endl;

}

}//end of for-loop

cout << "Press 9 to continue to the next screen. || Otherwise, press 1 to exit.\n";

cout << "Enter your response: ";

cin >> response;

cout << endl;

while (!(response == 1 || response == 9)){

cout << "Invalid selection. Re-enter your choice: ";

cin >> response;

cout << endl;

}

if (response == 1){

return; // exit program

}

else if(response == 9){

repWholesale();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*repWholesale\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*function #2 for reports menu\*/

void repWholesale(){

double total = 0;

int response;

// current date/time based on current system

time\_t now = time(0);

char arrEnd[] = {'\0'};

// convert now to string form

char \* dt = ctime(&now);

cout << "You selected Inventory Wholesale Value." << endl;

cout << "Here is the Inventory Wholesale Value Report on " << dt;

cout << "--------------------------------------------------------\n";

for (int index = 0; index < SIZE; index++)

{

if (strcmp(list[index].getTitle(), arrEnd) != 0) {

//display book information

cout << setw(2) << "" << "Title: " << list[index].getTitle() << endl;

cout << setw(2) << "" << "ISBN: " << list[index].getIBSN() << endl;

cout << setw(2) << "" << "Quantity-On-Hand: " <<list[index].getQty() << endl;

cout << fixed << showpoint << setprecision(2);

cout << setw(2) << "" << "Wholesale Cost: " << list[index].getWholesale() <<endl;

cout << endl;

}

}//end of for-loop

//compute sum of wholesale

total = totalSum(list,list, SIZE);

cout << fixed << showpoint << setprecision(2);

cout << "Total Wholesale Value: $" << total << "\n\n";

cout << "Press 9 to continue to the next screen. || Otherwise, press 1 to exit.\n";

cout << "Enter your response: ";

cin >> response;

cout << endl;

while (!(response == 1 || response == 9)){

cout << "Invalid selection. Re-enter your choice: ";

cin >> response;

cout << endl;

}

if (response == 1){

return; // exit program

}

else if(response == 9){

repRetail();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*repRetail\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*function #3 for reports menu\*/

void repRetail(){

double total = 0;

cout << "You selected Inventory Retail Value." << endl;

int response;

// current date/time based on current system

time\_t now = time(0);

char arrEnd[] = {'\0'};

// convert now to string form

char \* dt = ctime(&now);

cout << "Here is the Inventory Retail Value Report on " << dt;

cout << "--------------------------------------------------------\n";

for (int index = 0; index < SIZE; index++)

{

if (strcmp(list[index].getTitle(), arrEnd) != 0) {

//display book information

cout << setw(2) << "" << "Title: " << list[index].getTitle() << endl;

cout << setw(2) << "" << "ISBN: " << list[index].getIBSN() << endl;

cout << setw(2) << "" << "Quantity-On-Hand: " << list[index].getQty() << endl;

cout << fixed << showpoint << setprecision(2);

cout << setw(2) << "" << "Retail Price: " << list[index].getRetail() <<endl;

cout << endl;

}

}//end of for-loop

//compute sum of wholesale

total = totalSum(list, list, SIZE);

cout << fixed << showpoint << setprecision(2);

cout << "Total Retail Value: $" << total << "\n\n";

cout << "Press 9 to continue to the next screen. || Otherwise, press 1 to exit.\n";

cout << "Enter your response: ";

cin >> response;

cout << endl;

while (!(response == 1 || response == 9)){

cout << "Invalid selection. Re-enter your choice: ";

cin >> response;

cout << endl;

}

if (response == 1){

return; // exit program

}

else if(response == 9){

repQty();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*repQty\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*function #4 for reports menu\*/

void repQty(){

cout << "You selected Listing By Quantity." << endl;

int response;

// current date/time based on current system

time\_t now = time(0);

// convert now to string form

char \* dt = ctime(&now);

cout << "Here is the Listing by Quantity Report on " << dt;

cout << "--------------------------------------------------------\n";

//call sort function

QtySort(list, SIZE);

//user input to continue to next function

cout << "Press 9 to continue to the next screen. || Otherwise, press 1 to exit.\n";

cout << "Enter your response: ";

cin >> response;

cout << endl;

while (!(response == 1 || response == 9)){

cout << "Invalid selection. Re-enter your choice: ";

cin >> response;

cout << endl;

}

if (response == 1){

return; // exit program

}

else if(response == 9){

repCost();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*repCost\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*function #5 for reports menu\*/

void repCost(){

cout << "You selected Listing By Cost" << endl;

int response;

// current date/time based on current system

time\_t now = time(0);

// convert now to string form

char \* dt = ctime(&now);

cout << "Here is the Listing by Wholsesale Cost Report on " << dt;

cout << "--------------------------------------------------------\n";

//call sort function

costSort(list, SIZE);

//user input to continue to next function

cout << "Press 9 to continue to the next screen. || Otherwise, press 1 to exit.\n";

cout << "Enter your response: ";

cin >> response;

cout << endl;

while (!(response == 1 || response == 9)){

cout << "Invalid selection. Re-enter your choice: ";

cin >> response;

cout << endl;

}

if (response == 1){

return; // exit program

}

else if(response == 9){

repAge();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*repAge\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*stub function #6 for reports menu\*/

void repAge(){

cout << "You selected Listing By Age." << endl;

int response;

// current date/time based on current system

time\_t now = time(0);

// convert now to string form

char \* dt = ctime(&now);

cout << "Here is the Listing by Age Report on " << dt;

cout << "--------------------------------------------------------\n";

//call sort function

dateSort(list, SIZE);

//user input to continue to next function

cout << "Press 9 to continue to the next screen. || Otherwise, press 1 to exit.\n";

cout << "Enter your response: ";

cin >> response;

cout << endl;

while (!(response == 1 || response == 9)){

cout << "Invalid selection. Re-enter your choice: ";

cin >> response;

cout << endl;

}

if (response == 1){

return; // exit program

}

else if(response == 9){

reports();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Invmenu\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Implement user input for invmenu function via validation, while loops and display module menus\*/

void invmenu() {

const int SPACES = 7; //constant for formatting menu

int choice; //user input once validated

//display menu options repeatedly until user selects item #4

do

{

choice = temp.printInvMenu();

//display the user's menu selection

switch (choice){

case 1: lookupBook(); //call function

break;

case 2: addBook(); //call function

break;

case 3: editBook(); //call function

break;

case 4: deleteBook(); //call function

break;

case 5: cout << setw(SPACES -1) << "" << "You selected item " << choice << ". Exiting the program\n";

//main();

break;

default: cout << setw(SPACES -1) << "" << "That is an invalid choice.\n";

}

cout << "" << endl;

} while (choice != 5); //end of do-while loop

}

/\*Inventory database menu for searching inventory for a specific book title\*/

void lookupBook(){

const int LENGTH = 51;

char lookUp[LENGTH]; //hold user's input

cout << "You selected Look Up Book." << endl;

cout << "Please enter the title of the book, you wish to look up: ";

cin.ignore();

cin.getline(lookUp, LENGTH);

cout << endl;

found = temp.bookMatch(list, SIZE, lookUp);

if (found)

{

int index = searchFile(dataFile, "/Users/ecross/Desktop/booklist.txt", lookUp, SIZE);

if (index == -1){

cout << "The book was not found in the Serendepity Inventory." << endl;

return;

}else{

bookinfo(list,index);

}

}

invmenu();

}

/\*Inventory database menu for data about the books to the inventory\*/

void addBook(){

BookData book;

char t[51];

char isb[14];

char a[31];

char p[31];

char d[11];

int qty;

double sale, ret;

cout << "You selected Add Book." << endl;

if(openFileIn(dataFile, "/Users/ecross/Desktop/booklist.txt"))

{

cout << "File opened successfully.\n";

cout << "Now reading data from the file...\n\n";

addFileContents(dataFile);

dataFile.close();

cout << "\nDone.\n";

}

else

{

cout << "ERROR: Cannot open the file.\n";

}

int index = book.isEmpty(list, SIZE);

if(index != -1){

cout << "inventory empty at index " << index << endl;

//prompt user for book title

cin.ignore(256, '\n');

cout << "Set the book title: ";

cin.getline(t,51);

list[index].setTitle(t);

cout << endl;

cout << "Book title is: " << list[index].getTitle() <<endl;

//prompt user for isbn number

cin.ignore(256, '\n');

cout << "Set ISBN nummber: ";

cin.getline(isb,14);

list[index].setISBN(isb);

//strUpper(list[index].isbn);

cout << endl;

//prompt user for author's name

// cin.ignore(256, '\n');

cout << "Set author's name: ";

//cin.ignore(256, '\n');

cin.getline(a,31);

list[index].setAuthor(a);

// cout << endl;

//prompt user for publisher's name

cin.ignore(256, '\n');

cout << "Set publisher's name: ";

cin.getline(p,31);

list[index].setPub(p);

//strUpper(list[index].publisher);

cout << endl;

//prompt user for date (MM-DD-YY)

cout << "Set date book was added: ";

//cin.ignore();

cin.getline(d,11);

list[index].setDateAdded(d);

cout << endl;

//prompt user to quantity of book being added

cout << "Set quantity being added: ";

cin >> qty;

list[index].setQty(qty);

cout << endl;

//prompt user for wholesale cost for individual book

cout << "Set wholesale price: ";

cin >> sale;

list[index].setWholesale(sale);

cout << endl;

//prompt user for retail price for individual book

cout << "Set retail price: ";

cin >> ret;

list[index].setRetail(ret);

cout << endl;

}

writeToFile(dataFile, "/Users/ecross/Desktop/booklist.txt");

}

/\*function #3 for Inventory database menu\*/

void editBook(){

char t[51];

char isb[14];

char a[31];

char p[31];

char d[11];

int qty;

double sale, ret;

int count = -1;

const int LENGTH = 51;

char lookUp[LENGTH]; //hold user's input

//string title; //title of book, user wants to search for

int response; //user response to continue to edit book inventory

cout << "You selected Edit Book." << endl;

cout << "Please enter the title of the book, you wish to edit: ";

cin.ignore();

cin.getline(lookUp, LENGTH);

cout << endl;

found = temp.bookMatch(list, SIZE, lookUp);

if (found)

{

count = searchFile(dataFile, "/Users/ecross/Desktop/booklist.txt", lookUp, SIZE);

}

if (count != -1){

bookinfo(list, count);

cout << "Which fields would you wish to change?..." << endl;

do {

//show menu title

cout << setw(SPACES) << "" << "Serendipity Booksellers\n";

cout << setw(SPACES + 2) << "" << "Edit Information from the following fields\n\n";

//display book information

cout << setw(2) << "" << "1. ISBN " << endl;

cout << setw(2) << "" << "2. Title " << endl;

cout << setw(2) << "" << "3. Author " << endl;

cout << setw(2) << "" << "4. Publisher " << endl;

cout << setw(2) << "" << "5. Date Added " << endl;

cout << setw(2) << "" << "6. Quantity-On-Hand " << endl;

cout << setw(2) << "" << "7. Wholesale Cost " << endl;

cout << setw(2) << "" << "8. Retail Price " << endl;

cout << setw(2) << "" << "9. Exit " << endl;

cout << endl;

cout << setw(SPACES -1) << "" << "Enter Your Choice (1-9): ";

cin >> response; //enter user's answer

cin.ignore(256, '\n');

//user input validation

while (response < 1 || response > 9) {

cout << "Invalid answer. Please enter valid choice: ";

cin >> response;

cout << "" << endl; //newline

cin.ignore(256, '\n');

}

if (response == 1)

{

//prompt user for isbn

cout << "Set ISBN nummber: ";

cin.ignore(256, '\n');

cin.getline(isb,14);

list[count].setISBN(isb);

cout << endl;

cout << "isbn is now: " << list[count].getIBSN() << endl;

}

else if(response == 2)

{

//prompt user for title

cin.ignore(256, '\n');

cout << "Set the book title: ";

cin.getline(t,51);

list[count].setTitle(t);

cout << endl;

cout << "Book title is now: " << list[count].getTitle() << endl;

}

else if(response == 3)

{

//prompt user for author's name

cin.ignore(256, '\n');

cout << "Set author's name: ";

cin.getline(a,31);

list[count].setAuthor(a);

cout << endl;

cout << "Book author is now: " << list[count].getAuthor() << endl;

}

else if(response == 4)

{

//prompt user for publisher's name

cout << "Set publisher's name: ";

cin.ignore(256, '\n');

cin.getline(p,31);

list[count].setPub(p);

cout << endl;

cout << "publisher is now: " << list[count].getPub() << endl;

}

else if(response == 5)

{

//prompt user for date (MM-DD-YY)

cout << "Set date book was added: ";

cin.ignore();

cin.getline(d,11);

list[count].setDateAdded(d);

cout << endl;

cout << "date added is now: " << list[count].getDateAdded() << endl;

}

else if(response == 6)

{

//prompt user to quantity of book being added

cout << "Set quantity being added: ";

cin >> qty;

list[count].setQty(qty);

cout << endl;

cout << "quantity is now: " << list[count].getQty() << endl;

}

else if(response == 7){

//prompt user for wholesale cost for individual book

cout << "Set wholesale price: ";

cin >> sale;

list[count].setWholesale(sale);

cout << endl;

cout << "wholesale is now: $" << list[count].getWholesale() << endl;

}

else if(response == 8){

//prompt user for retail price for individual book

cout << "Set retail price: ";

cin >> ret;

list[count].setRetail(ret);

cout << endl;

cout << "retail price is now: $" << list[count].getRetail() << endl;

}

else if(response == 9){

cout << "end of editing book program.\n";

cout << "writing changes made to file...\n";

writeToFile(dataFile, "/Users/ecross/Desktop/booklist.txt");

return;

}

} while (response != 9); //end of do-while loop

}

else{

cout << "The book was not found in the Serendepity Inventory." << endl;

return;

}

}

/\*function #4 for Inventory database menu\*/

void deleteBook(){

int index = -1;

const int LENGTH = 51;

char lookUp[LENGTH]; //hold user's input

cout << "You selected Delete Book." << endl;

//string title; //title of book, user wants to search for

int answer; //user's input for deletion

cout << "Please enter the title of the book, you wish to delete: ";

cin.ignore();

cin.getline(lookUp, LENGTH);

cout << endl;

found = temp.bookMatch(list, SIZE, lookUp);

if (found) {

index = searchFile(dataFile, "/Users/ecross/Desktop/booklist.txt", lookUp, SIZE);

}

if (index != -1){

bookinfo(list, index);

cout << "Are you sure you want to delete this book from the inventory?..." << endl;

cout << "If yes, to delete: enter the number 1. | If no, to cancel: enter the number 2.\n";

cout << "Enter your choice: ";

cin >> answer;

cout << endl;

if(answer == 1){

list[index].removeBook(list, index);

cout << "Deletion program is complete. Writing to file the changes made." << endl;

writeToFile(dataFile, "/Users/ecross/Desktop/booklist.txt");

return;

}

if (answer == 2) {

cout << "Cancelling deletion. Exiting program.\n";

return;

}

}else{

cout << "The book was not found in the Serendepity Inventory." << endl;

return;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Bookinfo\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Implement user input for bookinfo function via validation, while loops and display module menus\*/

void bookinfo(BookData inv[], int index) {

const int SPACES = 7; //constant for formatting menu

//show menu title

cout << setw(SPACES) << "" << "Serendipity Booksellers\n";

cout << setw(SPACES + 2) << "" << "Book Information\n\n";

//display book information

cout << setw(2) << "" << "ISBN: " << inv[index].getIBSN() << endl;

cout << setw(2) << "" << "Title: " << inv[index].getTitle() << endl;

cout << setw(2) << "" << "Author: " << inv[index].getAuthor() << endl;

cout << setw(2) << "" << "Publisher: " << inv[index].getPub() << endl;

cout << setw(2) << "" << "Date Added: " << inv[index].getDateAdded() << endl;

cout << setw(2) << "" << "Quantity-On-Hand: " << inv[index].getQty() << endl;

cout << setw(2) << "" << "Wholesale Cost: " << inv[index].getWholesale() <<endl;

cout << setw(2) << "" << "Retail Price:" << inv[index].getRetail() << endl;

cout << endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Cashier\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*Implement user input for cashier function via validation, while loops and display module menus\*/

void cashier(){

string date; //date of book

int quantity[20]; //quantity of books

char isbn\_num[20][14]; //ISBN number

string title[20]; //title of book

float price[20]; //price of the book

float total =0;

float subtotal = 0;

float taxes = 0;

double bookTotal[20];

char anotherTransaction, answer; //user's response that holds Y or N input

const float PERCENT\_TAX = 0.06;

string i[20];

//char i[14];

int exit = 0;

do

{

subtotal = 0;

int count = 0;

cout << "Serendipity Book Sellers\n\n"; //display title

cout << " Cashier Module" << "\n\n"; //display date

//display and input date

cout << "Date: ";

cin >> date;

while ((exit != 1) && (count < 20)){

//display and input quanity of the book

cout << "Enter quantity of Book: ";

cin >> quantity[count];

cin.ignore(256, '\n');

//display and input isbn #

cout << "ISBN: ";

cin.getline(isbn\_num[count], 14);

int index = strSearchISBN(list, SIZE, isbn\_num[count]);

if (index != -1){

cout << "ISBN number was found!\n";

cout << "Title: " << list[index].getTitle() << "\n";

cout << "Retail Price: $" << list[index].getRetail() << "\n";

i[count] = list[index].getIBSN();

price[count] = list[index].getRetail();

title[count] = list[index].getTitle();

if (quantity[count] <= list[index].getQty()) {

list[index].setQty(list[index].getQty() - quantity[count]);

}

else{

cout << "There is not enough copies on-hand to fulfill purchase. Exiting program\n";

return;

}

}

else //when index = -1

{

do{

cout << "ISBN number could not be found\n";

cout << "Would you like to re-enter the ISBN number?...\n";

cout << "Please enter (Y=yes/ N=no): ";

cin >> answer;

//user input validation

while (!(answer == 'Y' || answer == 'y' || answer == 'N' || answer == 'n'))

{

cout << "Invalid answer. Please enter valid choice (Y=yes/ N=no): ";

cin >> answer;

cin.ignore();

cout << "" << endl; //newline

}

//return to main menu

if (answer == 'N' || answer == 'n') {

return;

}

//display and re-input isbn #

cout << "Re-Enter ISBN #: ";

cin.ignore(256, '\n');

cin.getline(isbn\_num[count],14);

cout << endl;

cout << "isbn: " << isbn\_num[count];

int index = strSearchISBN(list, SIZE, isbn\_num[count]);

//search isbn array

if (index != -1){

cout << "ISBN number was found!\n";

cout << "Title: " << list[index].getTitle() << "\n";

cout << "Retail Price: " << list[index].getRetail() << "\n";

//strcpy(i, isbn\_num[count]);

i[count] = list[index].getIBSN();

price[count] = list[index].getRetail();

title[count] = list[index].getTitle();

answer = 'n'; //exiting do-while loop

if (quantity[count] <= list[index].getQty()) {

list[index].setQty(list[index].getQty() - quantity[count]);

}

else{

cout << "There is not enough copies on-hand to fulfill purchase. Exiting program\n";

return;

}

}

else

{

cout << "No ISBN number was found...\n";

}

} while (answer == 'Y' || answer == 'y'); //end of do-while loop

}

cout << "\n";

//calculate individual book total

bookTotal[count] = quantity[count] \* price[count];

//calculate total price of books

subtotal += (quantity[count] \* price[count]);

count++;

cout << "enter 1 to calculate point-of-sales or enter 2 continue adding additional books: ";

cin >> exit;

cout << "exit entered is: " << exit << endl;

while(!(exit == 1 || exit == 2)){

cout << "invalid answer. Please enter valid choice: ";

cin >> exit;

cout << "exit entered is: " << exit << endl;

}

}//end of while loop

//calculate total amount of taxes

taxes = (subtotal \* PERCENT\_TAX);

//calculate total = subtotal + taxes

total = (subtotal + taxes);

cout << "Serendipity Book Sellers\n\n"; //display title

cout << "Date: " << date << "\n\n"; //display date

//cashier menu variables

cout << "Qty" << setw(2) << "" << "ISBN";

cout << setw(15) << "" << "Title" << setw(24) << "" << "Price";

cout << setw(6) << "" << "Total" << endl;

cout << setfill('-') << setw(75) << '\n';

cout << setprecision(2) << fixed;

for( int index = 0; index < SIZE; index++){

if(!(title[index].empty())){

//display user input

cout << setfill(' ')<< left << setw(4) << quantity[index] << ' ';

cout << setfill(' ') << left << setw(18) << isbn\_num[index] << ' ';

cout << setfill(' ')<< left << setw(28) << title[index] << ' ';

cout << setfill(' ')<< "$ " << left << setw(6) << price[index] << ' ';

cout << setfill(' ')<< " " << "$ " << left << setw(6) << bookTotal[index] << ' ';

cout << "" << endl; //newline

title[index] = "";

}

}

cout << "" << endl; //newline

//display variable of calculated total including subtotal, taxes and total

//cout << setprecision(2) << fixed;

cout << setfill(' ') << setw(12) << "" << "Subtotal " << ' ' << setw(44) << right << "$ " << setw(6) << subtotal << endl;

cout << setfill(' ') << setw(12) << "" << "Tax " << ' ' << setw(49) << right << "$ " << setw(6)<< taxes << endl;

cout << setfill(' ') << setw(12) << "" << "Total " << ' ' << setw(47)<<right << "$ " << setw(6) << total << endl;

cout << "" << endl; //newline

//display message

cout << "Thank You for Shopping at Serendipity!\n" << endl;

cout << "Is there another transaction to be processed?" << endl;

cout << " Please enter (Y=yes/ N=no): ";

cin >> anotherTransaction;

//user input validation

while (!(anotherTransaction == 'Y' || anotherTransaction == 'y' || anotherTransaction == 'N' || anotherTransaction == 'n')) {

cout << "Invalid answer. Please enter valid choice (Y=yes/ N=no): ";

cin >> anotherTransaction;

cin.ignore();

cout << "" << endl; //newline

}

exit = 0;

} while (anotherTransaction == 'Y' || anotherTransaction == 'y'); //end of do-while loop

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*repQty Sort Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void QtySort(BookData s[], int size)

{

char tempIsbn[14];

char tempTitle[51];

char tempAuthor[31];

char tempPub[31];

char tempDate[11];

char i[14];

char t[51];

char a[31];

char p[31];

char d[11];

int startScan, maxIndex, maxValue;

//string tempIsbn, tempTitle, tempAuthor, tempPub, tempDate;

double tempSale, tempRetail;

//outer for loop, to begin search through arrays, sorted by quantity array in descending order

for (startScan = 0; startScan < (size -1); startScan++)

{

maxIndex = startScan;

maxValue = s[startScan].getQty();

strcpy(tempIsbn, s[startScan].getIBSN());

strcpy(tempTitle,s[startScan].getTitle());

strcpy(tempAuthor,s[startScan].getAuthor());

strcpy(tempPub, s[startScan].getPub());

strcpy(tempDate,s[startScan].getDateAdded());

tempSale = s[startScan].getWholesale();

tempRetail = s[startScan].getRetail();

//inner loop

for(int index = startScan +1; index < size; index++)

{

if(s[index].getQty() > maxValue){

maxValue = s[index].getQty();

strcpy(tempIsbn, s[index].getIBSN());

strcpy(tempTitle,s[index].getTitle());

strcpy(tempAuthor,s[index].getAuthor());

strcpy(tempPub, s[index].getPub());

strcpy(tempDate,s[index].getDateAdded());

tempSale = s[index].getWholesale();

tempRetail =s[index].getRetail();

maxIndex = index;

}

}//end of inner for-loop

s[maxIndex].setQty(s[startScan].getQty());

strcpy(d, s[startScan].getDateAdded());

s[maxIndex].setDateAdded(d);

strcpy(i, s[startScan].getIBSN());

s[maxIndex].setISBN(i);

strcpy(t, s[startScan].getTitle());

s[maxIndex].setTitle(t);

strcpy(a, s[startScan].getAuthor());

s[maxIndex].setAuthor(a);

strcpy(p, s[startScan].getPub());

s[maxIndex].setPub(p);

s[maxIndex].setWholesale(s[startScan].getWholesale());

s[maxIndex].setRetail(s[startScan].getRetail());

//swap subscript values to sort in descending order

s[startScan].setQty(maxValue);

s[startScan].setISBN(tempIsbn);

s[startScan].setTitle(tempTitle);

s[startScan].setAuthor(tempAuthor);

s[startScan].setPub(tempPub);

s[startScan].setDateAdded(tempDate);

s[startScan].setWholesale(tempSale);

s[startScan].setRetail(tempRetail);

}//end of outer for-loop

char arrEnd[] = {'\0'};

//display sorted arrays

for (int index = 0; index < SIZE; index++)

{

if (strcmp(s[index].getTitle(), arrEnd) != 0) {

//display book information

cout << setw(2) << "" << "Title: " << s[index].getTitle() << endl;

cout << setw(2) << "" << "ISBN: " << s[index].getIBSN()<< endl;

cout << setw(2) << "" << "Quantity-On-Hand: " << s[index].getQty() << endl;

cout << endl;

}

}//end of for-loop

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*repCost Sort Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void costSort(BookData s[],int size)

{

int startScan, maxIndex, tempQty;

//temp holders for arrays

char tempIsbn[14];

char tempTitle[51];

char tempAuthor[31];

char tempPub[31];

char tempDate[11];

char i[14];

char t[51];

char a[31];

char p[31];

char d[11];

double maxValue, tempRetail;

//outer for loop, to begin search through arrays, sorted by quantity array in descending order

for (startScan = 0; startScan < (size -1); startScan++)

{

maxIndex = startScan;

maxValue = s[startScan].getWholesale();

strcpy(tempIsbn, s[startScan].getIBSN());

strcpy(tempTitle,s[startScan].getTitle());

strcpy(tempAuthor,s[startScan].getAuthor());

strcpy(tempPub, s[startScan].getPub());

strcpy(tempDate,s[startScan].getDateAdded());

tempQty = s[startScan].getQty();

tempRetail = s[startScan].getRetail();

//inner loop

for(int index = startScan +1; index < size; index++)

{

if(s[index].getWholesale() > maxValue){

maxValue = s[index].getWholesale();

strcpy(tempIsbn, s[index].getIBSN());

strcpy(tempTitle,s[index].getTitle());

strcpy(tempAuthor,s[index].getAuthor());

strcpy(tempPub, s[index].getPub());

strcpy(tempDate,s[index].getDateAdded());

tempQty= s[index].getQty();

tempRetail =s[index].getRetail();

maxIndex = index;

}

}//end of inner for-loop

s[maxIndex].setWholesale(s[startScan].getWholesale());

strcpy(d, s[startScan].getDateAdded());

s[maxIndex].setDateAdded(d);

strcpy(i, s[startScan].getIBSN());

s[maxIndex].setISBN(i);

strcpy(t, s[startScan].getTitle());

s[maxIndex].setTitle(t);

strcpy(a, s[startScan].getAuthor());

s[maxIndex].setAuthor(a);

strcpy(p, s[startScan].getPub());

s[maxIndex].setPub(p);

s[maxIndex].setQty(s[startScan].getQty());

s[maxIndex].setRetail(s[startScan].getRetail());

//swap subscript values to sort in descending order

s[startScan].setWholesale(maxValue);

s[startScan].setISBN(tempIsbn);

s[startScan].setTitle(tempTitle);

s[startScan].setAuthor(tempAuthor);

s[startScan].setPub(tempPub);

s[startScan].setDateAdded(tempDate);

s[startScan].setQty(tempQty);

s[startScan].setRetail(tempRetail);

}//end of outer for-loop

char arrEnd[] = {'\0'};

//display sorted arrays

for (int index = 0; index < SIZE; index++)

{

if (strcmp(s[index].getTitle(), arrEnd) != 0) {

//display book information

cout << setw(2) << "" << "Title: " << s[index].getTitle() << endl;

cout << setw(2) << "" << "ISBN: " << s[index].getIBSN() << endl;

cout << setw(2) << "" << "Quantity-On-Hand: " << s[index].getQty() << endl;

cout << setw(2) << "" << "Date Added: " << s[index].getDateAdded() << endl;

cout << endl;

cout << fixed << showpoint << setprecision(2);

cout << setw(2) << "" << "Wholesale Cost: $" << s[index].getWholesale() << endl;

cout << endl;

}

}//end of for-loop

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*repAge Sort Function\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void dateSort(BookData s[], int size)

{

int startScan, maxIndex, tempQty;

//temp holders for arrays

char tempIsbn[14];

char tempTitle[51];

char tempAuthor[31];

char tempPub[31];

char maxValue[11];

char i[14];

char t[51];

char a[31];

char p[31];

char d[11];

//string tempIsbn, tempTitle, tempAuthor, tempPub, maxValue;

double tempSale, tempRetail;

//outer for loop, to begin search through arrays, sorted by quantity array in descending order

for (startScan = 0; startScan < (size -1); startScan++)

{

maxIndex = startScan;

strcpy(maxValue, s[startScan].getDateAdded());

strcpy(tempIsbn, s[startScan].getIBSN());

strcpy(tempTitle,s[startScan].getTitle());

strcpy(tempAuthor,s[startScan].getAuthor());

strcpy(tempPub, s[startScan].getPub());

tempSale = s[startScan].getWholesale();

tempQty = s[startScan].getQty();

tempRetail = s[startScan].getRetail();

//inner loop

for(int index = startScan +1; index < size; index++)

{

if(strcmp(s[index].getDateAdded(),maxValue)> 0){

strcpy(maxValue,s[index].getDateAdded());

strcpy(tempIsbn, s[index].getIBSN());

strcpy(tempTitle,s[index].getTitle());

strcpy(tempAuthor,s[index].getAuthor());

strcpy(tempPub, s[index].getPub());

tempSale = s[index].getWholesale();

tempQty= s[index].getQty();

tempRetail =s[index].getRetail();

maxIndex = index;

}

}//end of inner for-loop

strcpy(d, s[startScan].getDateAdded());

s[maxIndex].setDateAdded(d);

strcpy(i, s[startScan].getIBSN());

s[maxIndex].setISBN(i);

strcpy(t, s[startScan].getTitle());

s[maxIndex].setTitle(t);

strcpy(a, s[startScan].getAuthor());

s[maxIndex].setAuthor(a);

strcpy(p, s[startScan].getPub());

s[maxIndex].setPub(p);

s[maxIndex].setWholesale(s[startScan].getWholesale());

s[maxIndex].setQty(s[startScan].getQty());

s[maxIndex].setRetail(s[startScan].getRetail());

//swap subscript values to sort in descending order

s[startScan].setDateAdded(maxValue);

s[startScan].setISBN(tempIsbn);

s[startScan].setTitle(tempTitle);

s[startScan].setAuthor(tempAuthor);

s[startScan].setPub(tempPub);

s[startScan].setWholesale(tempSale);

s[startScan].setQty(tempQty);

s[startScan].setRetail(tempRetail);

}//end of outer for-loop

char arrEnd[] = {'\0'};

//display sorted arrays

for (int index = 0; index < SIZE; index++)

{

if (strcmp(s[index].getTitle(), arrEnd) != 0) {

//display book information

cout << setw(2) << "" << "Title: " << s[index].getTitle() << endl;

cout << setw(2) << "" << "ISBN: " << s[index].getIBSN() << endl;

cout << setw(2) << "" << "Quantity-On-Hand: " << s[index].getQty()<< endl;

cout << setw(2) << "" << "Date Added: " << s[index].getDateAdded() << endl;

cout << endl;

}

}//end of for-loop

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*char search book title\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int strSearch(BookData list[], int size, char value[]){

int position = -1;

int index = 0;

int choice;

char \*strPtr = NULL;

strUpper(value);

cout << "looking up: " << value << endl;

cout << "Searching for match in inventory..." << endl;

for(index = 0; index < size; index++){

// cout << "title: " <<list[index] << endl;

strPtr = strstr(list[index].getTitle(), value);

if (strPtr != NULL){

cout << "title: " << list[index].getTitle()<< " at index " << index << endl;

cout << "Is this the book you were searching for?" << endl;

cout << "Enter 1 for YES | Enter 9 for NO\n";

cout << "Response: ";

cin >> choice;

cout << endl;

if (choice == 1){

position = index;

return position;

}

else if (choice == 9){

cout << "inventory search will resume...\n";

}

}

}

return position;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*char search by ISBN\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int strSearchISBN(BookData list[], int size, char value[]){

int position = -1;

int index = 0;

int choice;

char \*strPtr = NULL;

strUpper(value);

cout << "looking up: " << value << endl;

cout << "Searching for match in inventory..." << endl;

for(index = 0; index < size; index++){

cout << "title: " <<list[index].getTitle() << endl;

strPtr = strstr(list[index].getIBSN(), value);

if (strPtr != NULL){

cout << "ISBN #: " << list[index].getIBSN() << " at index " << index << endl;

cout << "Is this the book you were searching for?" << endl;

cout << "Enter 1 for YES | Enter 9 for NO\n";

cout << "Response: ";

cin >> choice;

cout << endl;

if (choice == 1){

position = index;

return position;

}

else if (choice == 9){

cout << "inventory search will resume...\n";

}

}

}

return position;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*char search book title boolean method \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool BookData::bookMatch(BookData list[], int size, char value[]){

bool titleFound = false;

int index = 0;

char \*strPtr = NULL;

strUpper(value);//convert search string to uppercase

cout << "Searching inventory for " << value << endl;

for(index = 0; index < size; index++){

strPtr = strstr(list[index].getTitle(), value);

cout << list[index].getTitle() << endl;

//cout << strPtr << endl;

if (strPtr != NULL){

//cout << "title: " << list[index].getTitle()<< " at index " << index << endl;

cout << "A possible match has been found." << endl;

cout << "Continuing a more in-depth search...";

titleFound = true;

return titleFound;

}

}

cout << "no match found." <<endl;

return titleFound;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Display Main Menu using class objects \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int BookData::printMainMenu()

{

const int SPACES = 7; //constant for formatting menu

int choice; //allows user to enter choice

//display menu options repeatedly until user selects item #4

//show menu title

cout << endl;

cout << setw(SPACES-3) << "" << "Serendipity Booksellers\n";

cout << setw(SPACES + 5) << "" << "Main Menu\n\n";

//show menu options

cout << setw(SPACES -1) << "" << "1. Cashier Module" << endl;

cout << setw(SPACES -1) << "" << "2. Inventory Database Module" << endl;

cout << setw(SPACES -1) << "" << "3. Report Module" << endl;

cout << setw(SPACES -1) << "" << "4. Exit\n" << endl;

cout << setw(SPACES -1) << "" << "Enter Your Choice (1-4): ";

cin >> choice; //enter user's answer

cout << "" << endl;

//input validation

while(choice < 1 || choice > 4)

{

cout << setw(SPACES -1) << "" << "Please enter a number in the range of 1-4.\n\n"; //error message

//user re-enter's menu choice

cout << setw(SPACES -1) << "" << "Enter Your Choice (1-4): ";

cin >> choice; //enter user's answer

cout << "\n";

}

return choice;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Display Report Menu using class objects \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int BookData::printReportMenu()

{

//show menu title

int choice;

cout << setw(SPACES) << "" << "Serendipity Booksellers\n";

cout << setw(SPACES + 8) << "" << "Reports\n\n";

//show menu options

cout << setw(SPACES -1) << "" << "1. Inventory Listing" << endl;

cout << setw(SPACES -1) << "" << "2. Inventory Wholesale Value" << endl;

cout << setw(SPACES -1) << "" << "3. Inventory Retail Value" << endl;

cout << setw(SPACES -1) << "" << "4. Listing by Quantity" << endl;

cout << setw(SPACES -1) << "" << "5. Listing by Cost" << endl;

cout << setw(SPACES -1) << "" << "6. Listing by Age" << endl;

cout << setw(SPACES -1) << "" << "7. Return to Main Menu\n" << endl;

cout << setw(SPACES -1) << "" << "Enter Your Choice (1-7): ";

cin >> choice; //enter user's answer

//input validation

while(choice < 1 || choice > 7)

{

cout << setw(SPACES -1) << "" << "Please enter a number in the range of 1-7.\n\n"; //error message

//user re-enter's menu choice

cout << setw(SPACES -1) << "" << "Enter Your Choice (1-7): ";

cin >> choice; //enter user's answer

cout << "\n";

}

return choice;

}

int BookData::printInvMenu()

{

int choice;

//show menu title

cout << setw(SPACES) << "" << "Serendipity Booksellers\n";

cout << setw(SPACES + 2) << "" << "Inventory Database\n\n";

//show menu options

cout << setw(SPACES -1) << "" << "1. Look Up a Book" << endl;

cout << setw(SPACES -1) << "" << "2. Add a Book" << endl;

cout << setw(SPACES -1) << "" << "3. Edit a Book's Record" << endl;

cout << setw(SPACES -1) << "" << "4. Delete a Book" << endl;

cout << setw(SPACES -1) << "" << "5. Return to Main Menu\n" << endl;

cout << setw(SPACES -1) << "" << "Enter Your Choice (1-5): ";

cin >> choice; //enter user's answer

//input validation

while(choice < 1 || choice > 5)

{

cout << setw(SPACES -1) << "" << "Please enter a number in the range of 1-5.\n\n"; //error message

//user re-enter's menu choice

cout << setw(SPACES -1) << "" << "Enter Your Choice (1-5): ";

cin >> choice; //enter user's answer

cout << "\n";

}

return choice;

}