CIS22B Final Project
Team: 3
Members: Barr, Brandon, Noah, and Rohan

Requirement Analysis:

1. Program can store up to 25 unique books from text file (inventtory). Inventory file updated when books are added/removed or edited. Various sales reports are displayed.
2. **Cashier module:** adds books to cart for purchase and updates inventory on checkout,
3. **Inventory:** keeps track of 25 unique books each with the following information fields: ISBN,Title, Author, Publisher, Date Added, Quantity, Wholesale Cost, Retail Price; Books in inventory can be: looked up, added, removed, or edited.
4. **The Report Module:** Displays reports for the following: Inventory List, Inventory Wholesale Value, Inventory Retail Value, List by Quantity, List by Cost, List by Age.

5. Main OOP concepts(Classes, Inheritance, Polymorphism, Friends): demonstrated by the three classes **Inventory, BookPile, book.** Where **BookPile** is the child of **book** and class inventory is a friend of **BookPile**. Polymorphism is used by using **book** pointer on its child **BookPile** to use **book** "<<" overload instead of **BookPile** "<<" overload.
    i. Polymorphism Demonstrated in the following functions:
        1. **Inventory::printInventorReport()**; ReportModule.cpp
        2. **Inventory::inventoryRetail (std::string name)**; ReportModule.cpp
        3. **Inventory::inventoryWholesale (std::string name)**; ReportModule.cpp
        4. **Inventory::inventoryQuantity(std::string name)**; ReportModule.cpp
        5. **Inventory::inventoryAge(std::string name)**; ReportModule.cpp
6. Operator Overloads:
    a. book
        i. friend operator<<(output : std::ostream&, toPrint: book&) : std::ostream&
    b. BookPile
        i. operator+(right_int : int) : BookPile
        ii. operator++(: int) : BookPile
        iii. operator+() : BookPile
        iv. operator-(right_int : int) : BookPile
        v. operator--(: int) : BookPile
        vi. operator--() : BookPile
        vii. friend operator<<(output : std::ostream&, toPrint: BookPile&) : std::ostream&

Classes Information:

book:

Name: book
Member Variables: * ISBN : int * title : std::string * author : std::string * publisher : std::string * dateAdded[3] : int * wholesale_cost : double * retail_price : double
Member Functions: + book() : + book(nISBN : int, ntitle : std::string, nauthor : std::string, npublisher : std::string, month : int, day : int, year : int, nwhole_sale_cost : double, nretail_price : double) : + getISBN() : int + getTitle() : std::string + getAuthor() : std::string + getPublisher() : std::string + getDate() : int* + getWholeSaleCost() : double + getRetailPrice() : double + setISBN(number : int) : void + setName(name : std::string) : void + setAuthor(title : std::string) : void + setPublisher(publisher : std::string) : void + setDate(month : int, day : int, year : int) : void + setWholeSaleCost(nCost : double) : void + setRetailPrice(nCost : double) : void + friend operator<<(output : std::ostream&, toPrint: book&) : std::ostream&

Book Overloads:

```
/*****************************************************************************/
//
// Function: std::ostream& operator<<(std::ostream& output, BookPile& toPrint)
//
// Parameters: none
// (Inputs)
//
// Outputs: some of the book's members variables printed in the format:
//          ISBN: title
//
// Returns: output
//
// Description: This function overloads the "<<". When used it prints
// some of the book's information into the chosen output.
/*****************************************************************************/
```

BookPile:

Name: BookPile

Member Variables:
- ISBN : int
- title : std::string
- author : std::string
- publisher : std::string
- dateAdded[3] : int
- wholesale_cost : double
- retail_price : double
- quantity : int

Member Functions:
+ book() :
+ book(nISBN : int, ntitle : std::string, nauthor : std::string, npublisher : std::string, month : int, day : int, year : int, nwhole_sale_cost : double, nretail_price : double) :
+ getISBN() : int
+ getTitle() : std::string
+ getAuthor() : std::string
+ getPublisher() : std::string
+ getDate() : int*
+ getWholeSaleCost() : double
+ getRetailPrice() : double
+ getQuantity() : int
+ setISBN(number : int) : void
+ setName(name : std::string) : void
+ setAuthor(title : std::string) : void
+ setPublisher(publisher : std::string) : void
+ setDate(month : int, day : int, year : int) : void
+ setWholeSaleCost(nCost : double) : void
+ setRetailPrice(nCost : double) : void
+ setQuantity(quantity : int) : void
+ operator+(right_int : int) : BookPile
+ operator++(: int) : BookPile
+ operator+() : BookPile
+ operator-(right_int : int) : BookPile

+ operator--(: int) : BookPile
+ operator--() : BookPile
+ friend operator<<(output : std::ostream&, toPrint: BookPile&) : std::ostream&

BookPile Overloads:

```
/*****************************************************************************/
// Function: BookPile BookPile::operator+(int right_int)
//
// Parameters: int right_int - integer amount to add to the quantity
// (Inputs)
//
// Outputs:    none
//
// Returns:    BookPile with quantity increased by right_int
//
// Description: This function overloads the "+" operator with an int. When
// used it increases the quantity of the BookPile by the int to the right of
// the "+" operator;
/*****************************************************************************/
/*****************************************************************************/
// Function: BookPile BookPile::operator+(int right_int)
//
// Parameters: none
// (Inputs)
//
// Outputs:    none
//
// Returns:    BookPile with quantity incremented by 1
//
// Description: This function overloads the "++" aka pre-increment operator.
// When used it increments the quantity of the BookPile by 1;
/*****************************************************************************/




/*****************************************************************************/
```

```
// Function: BookPile BookPile::operator-(int right_int)
//
// Parameters: int right_int - integer amount to remove from the quantity
// (Inputs)
//
// Outputs:    none
//
// Returns:    BookPile with quantity decreased by right_int
//
// Description: This function overloads the "-" operator with an int. When
// used it decreases the quantity of the BookPile by the int to the right of
// the "-" operator;
/*****************************************************************************/
/*****************************************************************************/
// Function: BookPile BookPile::operator--(int)
//
// Parameters: none
// (Inputs)
//
// Outputs:    none
//
// Returns:    BookPile with quantity decremented by 1
//
// Description: This function overloads the "++" aka post-decrement operator.
// When used it decrements the quantity of the BookPile by 1;
/*****************************************************************************/
/*****************************************************************************/
// Function: BookPile BookPile::operator--()
//
// Parameters: none
// (Inputs)
//
// Outputs:    none
//
// Returns:    BookPile with quantity decremented by 1
//
// Description: This function overloads the "++" aka pre-decrement operator.
// When used it decrements the quantity of the BookPile by 1;
/*****************************************************************************/



/*****************************************************************************/
```

```
// Function: std::ostream& operator<<(std::ostream& output, BookPile& toPrint)
//
// Parameters: none
// (Inputs)
//
// Outputs: some of the BookPile’s members variables printed in the format:
//          ISBN: title (retail_price) X quantity
//
// Returns: output
//
// Description: This function overloads the "<<". When used it prints
// some of the BookPile’s information into the chosen output.
/*****************************************************************************/
```

Inventory:

Name: Inventory
Member Variables: - numOfBooks : const int - books[25] : BookPile
Member Functions: + Inventory() + printInventory() : void + readFile() : void + getBook(int) : + setBook(int,BookPile) : void + addBook(BookPile b) : void + writeFile() : void + sortBooks() : void + searchBooks(BookPile) : int + searchBookByISBN(int) : int + removeBook(BookPile) : void

Inventory Functions/Overloads:

void Inventory::inventoryWholesale(std::string name)
//prints out a list displaying whole sale price (empty books not displayed)

```
/*****************************************************************************/
//
// Function: inventoryWholesale
//
// Parameters: std::string name
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
//
// Description:
// Prints out whole sale Price, ISBN and Title of all books in inventory ignores any empty entrys form printing.
```

// Demonstrates polymorphism by useing BookPile's parent books << operator overload instead of it's own class.
//
// entering ifrent name changes defalt title of print out. ISBN formated to print 4 digits leading 0's and price to print out 2 decimal places
//
// prints total whole sale price
/***/

Inventory::printInventoryReport():
//prints out a Inventory List for Report Screen (empty books not displayed)
/***/
//
// Function: inventoryWholesale
//
// Parameters: std::string name
// (Inputs)
//
// Outputs: none
//
// Returns: void
//
// Description:
// Prints out ISBN and Title of all books in inventory ignores any empty entrys form printing.
// Demonstrates polymorphism by useing BookPile's parent books << operator overload instead of it's own class.
//
// ISBN formated to print 4 digits leading 0's and price to print out 2 decimal places
//
/***/

void Inventory::inventoryRetail(std::string name):
//prints out a list displaying retail price (empty books not displayed)
/***/
//
// Function: inventoryRetail
//
// Parameters: std::string name
// (Inputs)
//
// Outputs: none
//
// Returns: void

```
//
// Description:
// Prints out retail Price, ISBN and Title of all books in inventory ignores any empty entrys form printing.
// Demonstrates polymorphism by useing BookPile's parent books << operator overload instead of it's own class.
//
// entering difrent name changes defalt title of print out. ISBN formated to print 4 digits leading 0's and price to print out 2 decimal places
//
// prints total retail price
/*****************************************************************************/

void Inventory::inventoryRetail(std::string name):
//prints out a list displaying retail price (empty books not displayed)
/*****************************************************************************/
//
// Function: inventoryRetail
//
// Parameters: std::string name
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
//
// Description:
// Prints out retail Price, ISBN and Title of all books in inventory ignores any empty entrys form printing.
// Demonstrates polymorphism by useing BookPile's parent books << operator overload instead of it's own class.
//
// entering difrent name changes defalt title of print out. ISBN formated to print 4 digits leading 0's and price to print out 2 decimal places
//
// prints total retail price
/*****************************************************************************/

void Inventory::inventoryQuantity(std::string name):
//prints out a list displaying number of books (empty books not displayed)
/*****************************************************************************/
//
// Function: inventoryQuantity
```

```
//
// Parameters: std::string name
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
//
// Description:
// Prints out Quantity, ISBN and Title of all books in inventory ignores any empty entrys form printing.
// Demonstrates polymorphism by useing BookPile's parent books << operator overload instead of it's own class.
//
// entering difrent name changes defalt title of print out. ISBN formated to print 4 digits leading 0's and price to print out 2 decimal places
//
// displays sold out if book quantity is 0
/*****************************************************************************/

void Inventory::inventoryAge(std::string name):
//prints out a list displaying date added (empty books not displayed)
/*****************************************************************************/
//
// Function: inventoryAge
//
// Parameters: std::string name
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
//
// Description:
// Prints out date added, ISBN and Title of all books in inventory ignores any empty entrys form printing.
// Demonstrates polymorphism by useing BookPile's parent books << operator overload instead of it's own class.
//
// entering difrent name changes defalt title of print out. ISBN formated to print 4 digits leading 0's and price to print out 2 decimal places
//
/*****************************************************************************/
```

void Inventory::sortByWholeSaleCost():
//Sorts by wholesale price then prints out a list displaying books and wholesale price(empty books not displayed)

```
/*****************************************************************************/
//
// Function: Inventory::sortByWholeSaleCost()
//
// Parameters: none
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
//
// Description:
// function of the inventory class that sort list by wholesale cost useing select sort; highest to lowest.
//  calls inventory class function "inventoryWholesale" to display results
//
/*****************************************************************************/
```

void Inventory::sortByQuantity():
//Sorts by Quantity then prints out a list displaying books and Quantity(empty books not displayed)

```
/*****************************************************************************/
//
// Function: Inventory::sortByQuantity()
//
// Parameters: none
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
//
// Description:
// function of the inventory class that sort list by quantity useing select sort; highest to lowest.
// calls inventory class function "inventoryQuantity()" to display results
//
/*****************************************************************************/
```

void Inventory::sortByAge():

//Sorts by date added then prints out a list displaying books and date added(empty books not displayed)
/***/
//
// Function: Inventory::sortByQuantity()
//
// Parameters: none
// (Inputs)
//
// Outputs: none
//
// Returns: void
//
// Description:
// function of the inventory class that sort list by date added useing select sort; oldest to newest.
// calls inventory class function "inventoryAge()" to display results
//
/***/

writeFile()
//Stores the books in the Inventory object in a txt file.
/***/
//
// Function: writeFile
//
// Parameters: none
// (Inputs)
//
// Outputs: none
//
// Returns: void
//
// Description:
// Stores the books in the Inventory object in the "output.txt" file.
Pseudo-code:
for(Loop through book array){
	if(isbn is not 0){
		Store each of the individual values such as isbn, author, title in the file
	}
}
/***/

printInventory()

```
//prints out the Inventory
/*****************************************************************************/
//
// Function: printInventory
//
// Parameters: none
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
//
// Description:
// Prints out ISBN, title, and quantity of all books in inventory
Pseudo-code:
for(loop through books array){
        if(Title is not empty)
                Print the book
        Else
                Print empty book shelf
}
/*****************************************************************************/
```

readFile()

```
//Reads the file of books and stores it in an array.
/*****************************************************************************/
//
// Function: readFile
//
// Parameters: none
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
//
// Description:
// Reads the txt file "books.txt" and stores the information in the inventory array.
//
Pseudo-code:
file.open(filename here){
        while(file has more lines){
                Store ISBN, author, publisher, and all other values into BookPile object
```

```
                Add BookPile object into books array
        }
}
/*****************************************************************************/
```

getBook(int)

```
//Returns the BookPile object stored in the index the user enters.
/*****************************************************************************/
//
// Function: getBook
//
// Parameters: int bookNumber
// (Inputs)
//
// Outputs:    none
//
// Returns:    BookPile
//
// Description:
// Returns the BookPile object that is stored in the index the user enters.
//
Pseudo-code:
return books[index];
/*****************************************************************************/
```

setBook(int, BookPile)

```
//Setter for books array
/*****************************************************************************/
//
// Function: setBook
//
// Parameters: int booknumber, BookPile book
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
//
// Description:
// Lets the user enter an index of the array and a BookPile objects and stores
// the BookPile object in that index of the array.
//
Pseudo-code:
```

```
books[bookNumber]=book;
/*****************************************************************************/

addBook();
//Adds a book the user enters to the inventory.
/*****************************************************************************/
//
// Function: addBook
//
// Parameters: BookPile b
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
//
// Description:
// Checks to see if the book is available in the inventory. If it is, then adds one
// to the quantity of that book. If not, it will add the book to inventory. If the
// inventory is full, it will print a message saying there is no more space in
// the inventory.
//
Pseudo-code:
for(loop through array){
        if(isbn of b is equal to isbn of a book in array){
                Then add one to the quantity of that BookPile object
        }
}
if(b is not in books array){
        if(check when books has an empty slot){
                Stores the BookPile object in empty slot
        }
}
if(no space in inventory){
        Print message no space
}
/*****************************************************************************/

sortBooks()
//Sorts the books in the array.
/*****************************************************************************/
//
// Function: sortBooks()
```

```
//
// Parameters: none
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
//
// Description:
// Uses a for loop to loop through the books array and sorts the books array
// in alphabetical order.
//
Pseudo-code:
for(loop through array){
        for(){
                Check to see which book is greater in terms of alphabetical order and swap
        }

}
/*****************************************************************************/

searchBooks(BookPile)
//Searches the books array
/*****************************************************************************/
//
// Function: searchBooks
//
// Parameters: BookPile b
// (Inputs)
//
// Outputs:    none
//
// Returns:    int
//
// Description:
// Searches the books array. If an isbn number matches the users BookPile objects
// isbn number then it returns the index of where that BookPile object is located
// in the array. If there is no match in the array it will return -1.
//
Pseudo-code:
for(loop through book array){
        if(ISBN matches){
                return index of book
```

```
        }
}
/*****************************************************************************/
```

searchBookByISBN(int isbn)

```
//Searches the books array
/*****************************************************************************/
//
// Function: searchBookByISBN
//
// Parameters: int isbn
// (Inputs)
//
// Outputs:    none
//
// Returns:    int
//
// Description:
// Searches the books array while taking the isbn as a parameter. If the isbn
// number matches one of the BookPile objects isbn numbers in the book array,
// it will return the index of that BookPile object. If no match is found,
// this function will return -1.
//
Pseudo-code:
for(loop through array){
        if(ISBN matches){
                Return index of books array
        }
}
/*****************************************************************************/
```

removeBook(BookPile)

```
//Removes a BookPile object in the book array
/*****************************************************************************/
//
// Function: removeBook
//
// Parameters: BookPile b
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
```

```
//
// Description:
// Removes the BookPile object b that the user enters in the function parameter.
// If the quantity of a book is 0, an empty BookPile object will be stored in
// place of the book.
//
Pseudo-code:
for(loop through books array){
        if(quantity of books is not 0){
                quantity--
        }
        else{
                Store empty book in that part of array
        }
}
/*****************************************************************************/
```

Program pseudo-code:

```
int main(){
        while(user wants program to run){
                displayMenu();
                takeInUserInput();

                switch(the user's input){
                        case 1: //cashier mode
                                Create cart;
                                while(the user is in cashier mode){
                                        displayMenuForCashier();
                                        takeInUserInput();
                                        switch(the user's input){
                                                case 1: //Add book to cart
                                                        Prompt user for isbn;
                                                        Prompt user for quantity;
                                                        Search for that isbn in bookstore;
                                                        add quantity of that book to cart;
                                                        Remove quantity of that book from
                                                        bookstore;
                                                        break;
                                                case 2: //Remove book from cart
                                                        Prompt user for isbn;
                                                        Prompt user for quantity;
                                                        Search for that isbn in cart;
                                                        add quantity of that book to bookstore;
                                                        Remove quantity of that book from cart;
                                                        break;
                                                case 3: //checkout
                                                        Write updates of BookStore to file
                                                        Print receipt
                                                        End cashier loop
                                                        Return to main menu
                                                        break;
                                                case 4: //return to main menu
                                                        End cashier loop
                                                        Return to main menu
                                                        break;
                                        }
                                }
                                break;
                        case 2: //inventory mode
```

```
        while(the user is in cashier mode){
            displayMenuForInventory();
            takeInUserInput();
            switch(the user's input){
                case 1: //select a new book
                    Prompt user for isbn;
                    Search for that isbn in bookstore;
                    Return information of that book;
                    break;
                case 2: //add a new book
                    Prompt user for isbn;
                    Make sure user enters an isbn that doesn't
                    already exist in the bookstore;
                    Prompt user for the rest of the information of
                    the book they wish to add;
                    Add the new book to inventory;
                    Write updates of BookStore to file;
                    break;
                case 3: //delete existing book
                    Prompt user for isbn;
                    Search for that isbn in bookstore;
                    Delete book from BookStore;
                    Write updates of BookStore to file;
                    break;
                case 4: //edit existing book
                    Prompt user for isbn;
                    Search for that isbn in bookstore;
                    Prompt user for the rest of the information of
                    the book they wish to edit;
                    Edit that book;
                    Write updates of BookStore to file;
                    break;
                case 5: //return to main menu
                    End inventory loop;
                    Return to main menu;
                    break;
            }
        }
        break;
case 3: //query report
        while(the user is in cashier mode){
            displayMenuForQueryReport();
            takeInUserInput();
```

```
                                        switch(the user's input){
                                                case 1: //inventory list
                                                        Print inventory list;
                                                        break;
                                                case 2: //inventory wholesale value list
                                                        Print inventory list sorted by wholesale
                                                        value;
                                                        break;
                                                case 3: //inventory retail value list
                                                        Print inventory list sorted by retail value;
                                                        break;
                                                case 4: //list by quantity
                                                        Print inventory list sorted by quantity;
                                                        break;
                                                case 5: //list by cost
                                                        Print inventory list sorted by cost;
                                                        break;
                                                case 6: //list by age
                                                        Print inventory list sorted by age
                                                        break;
                                                case 7: //return to main menu
                                                        End report loop
                                                        Return to main menu
                                                        break;
                                        }
                                break;
                        case 4: //end program
                                break;
                        default: //tell user their input was invalid
                                break;
                }
        return 0;
}
```

Report Mode Functions:

```
//Report mode switch controls sub windows that user enters and leaves.
/*****************************************************************************/
//
// Function: queryReport
//
// Parameters: bool& run, char& mode,Inventory* BookStore, int shelfSize, char& screen
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
//
// Description:
// displays report screen. calls user_navigator to check for valid input and control navigation using mode (to move between modes) and user command (to move within mode).
// Screen clears and reprints each time user enters a command. Bookstore inventory passed along with other variables linking module to main menu.
//
// submenues:
//   1. print inventory list
//   2. print inventory wholesale value list
//   3. print invetory retail value list
//   4. print list by quantity
//   5. print list by cost
//   6. print list by age
//   7. exits to main menu
//
/*****************************************************************************/
```

User navigator: (for main menu and report module)

```
//User navigation for main menu and report module
/*****************************************************************************/
//
// Function: user_navigator
//
// Parameters: char& screen, int& userComand,int numberOfuserOptions
// (Inputs)
//
// Outputs:    none
//
// Returns:    bool
```

```
//
// Description:
// displays prompt for user comand used to switch between screens and comands within
screen.only used in main menu and report screen.
// checks if user input is valid and if so changes screen or user comand value. number of user
options value is used to check if valid comand is given
//
/*****************************************************************************/
```

Functions

modeDisplay:

```
//prints mode on screen
/*****************************************************************************/
//
// Function: modeDisplay
//
// Parameters: char screen
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
//
// Description:This displayes the screen the user is on
//
/*****************************************************************************/
```

reportComandList():

```
//prints all comands user can use on this remort module
/*****************************************************************************/
//
// Function: reportComandList()
//
// Parameters: none
// (Inputs)
//
// Outputs:    none
//
// Returns:    void
//
// Description:
//          Report Screen Comand List:
//          Prints out all valid user inputs for screen 3
```

```
//      1.      [1]: Inventory List: A list of information on all books in the inventory.
//      2.      [2]: Inventory Wholesale  Value List:
//      3.      [3]: Inventory  Retail  Value
//      4.      [4]: List by Quantity
//      5.      [5]: List by Cost
//      6.      [6]: List  by  Age
//
//
/*****************************************************************************/
```

BookStore File:

Format:

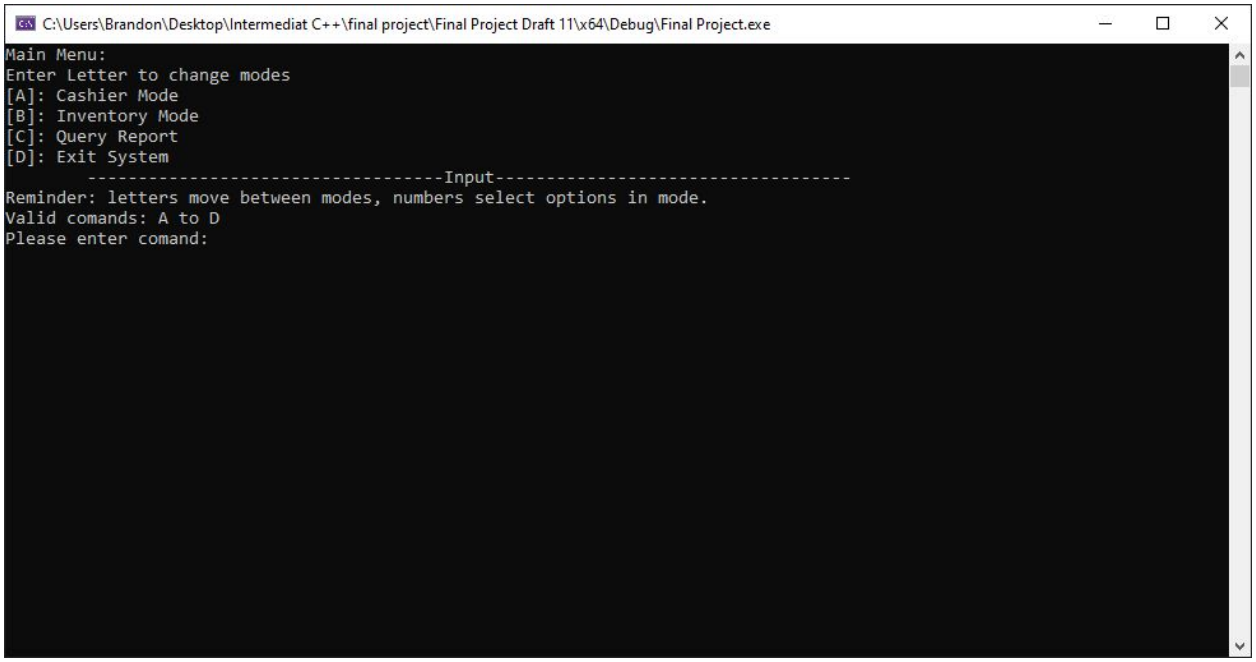
ISBN Title, Author, Publisher, Month.Day.Year, whole_sale_cost retail_cost quantity

Example:

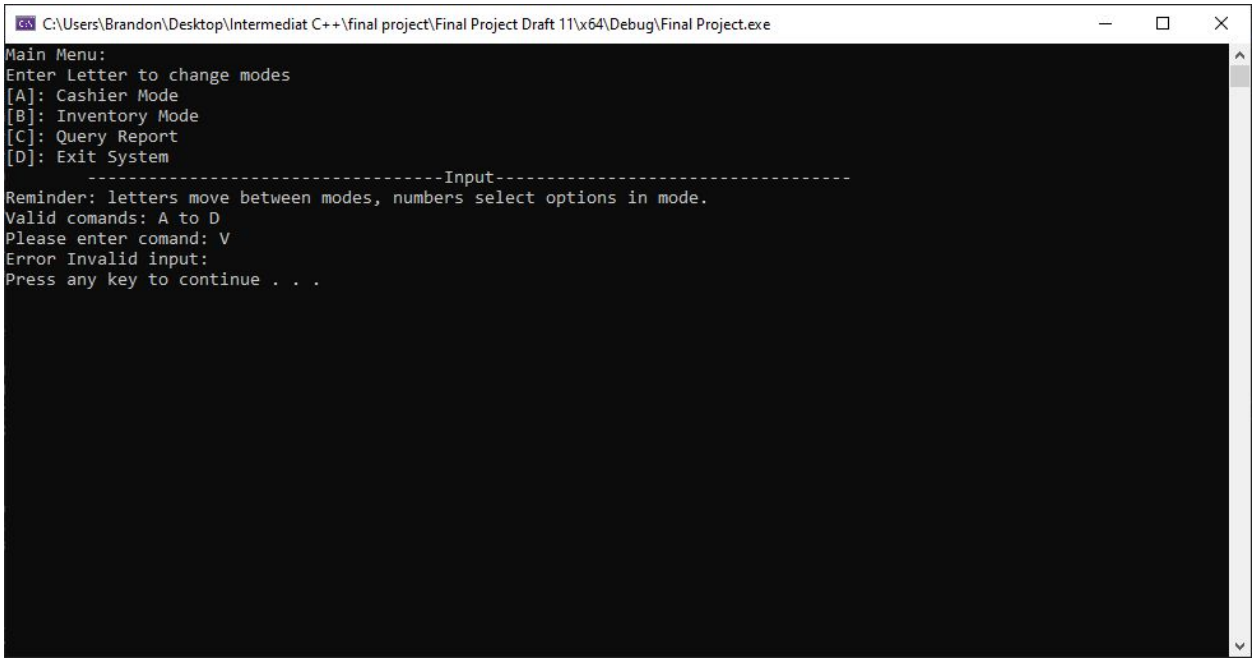
1006 Peter Pan, nothing, Neverland, 6.12.2014, 22.23 3 1
2004 Evil Dead the Musical, Ash, mystry threater, 7.12.1987, 22.21 3 1
1002 Back to the future 2, Doc Brown, Delorian, 1.1.2024, 12.21 4 1
1003 Back to the future 3, Doc Brown, Delorian, 11.1.1874, 12.11 4 0
1001 Back to the future 1, Doc Brown, Delorian, 11.2.1974, 12.11 4 0
1005 Test Case Book, Ananymus, publisher, 10.2.2014, 12.11 4 0
6660 Necronomicon, HP lovecraft, cathulu, 11.21.2014, 100 5 1

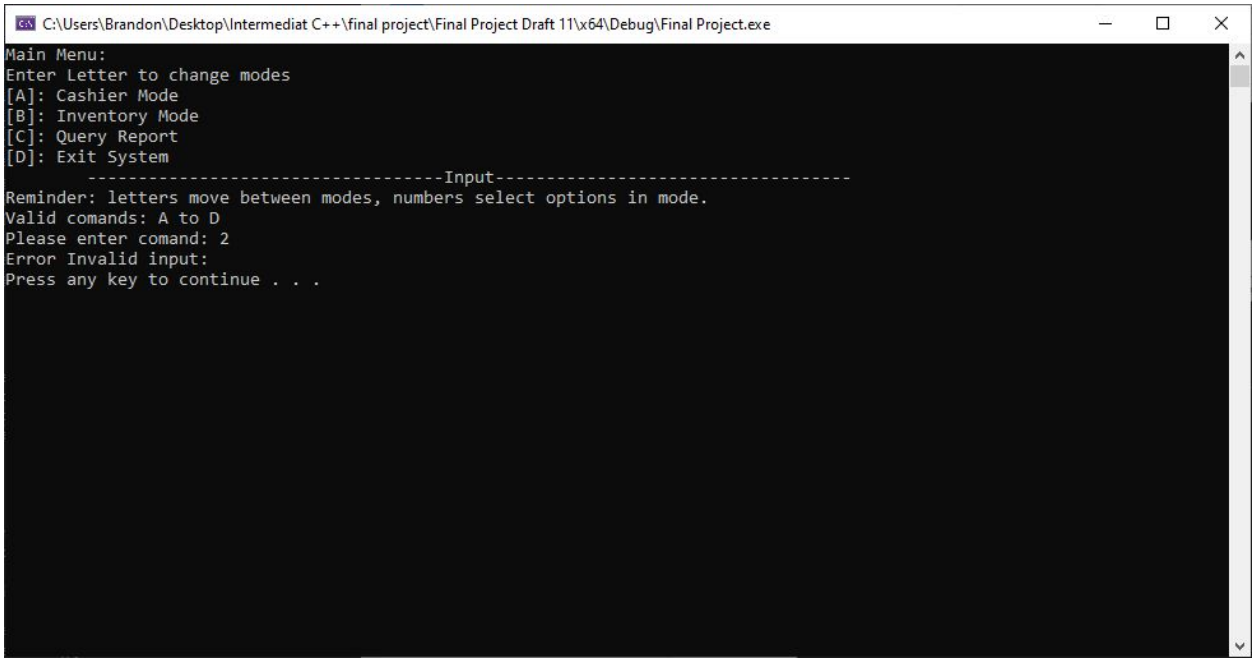
Screenshots:

Main menu:

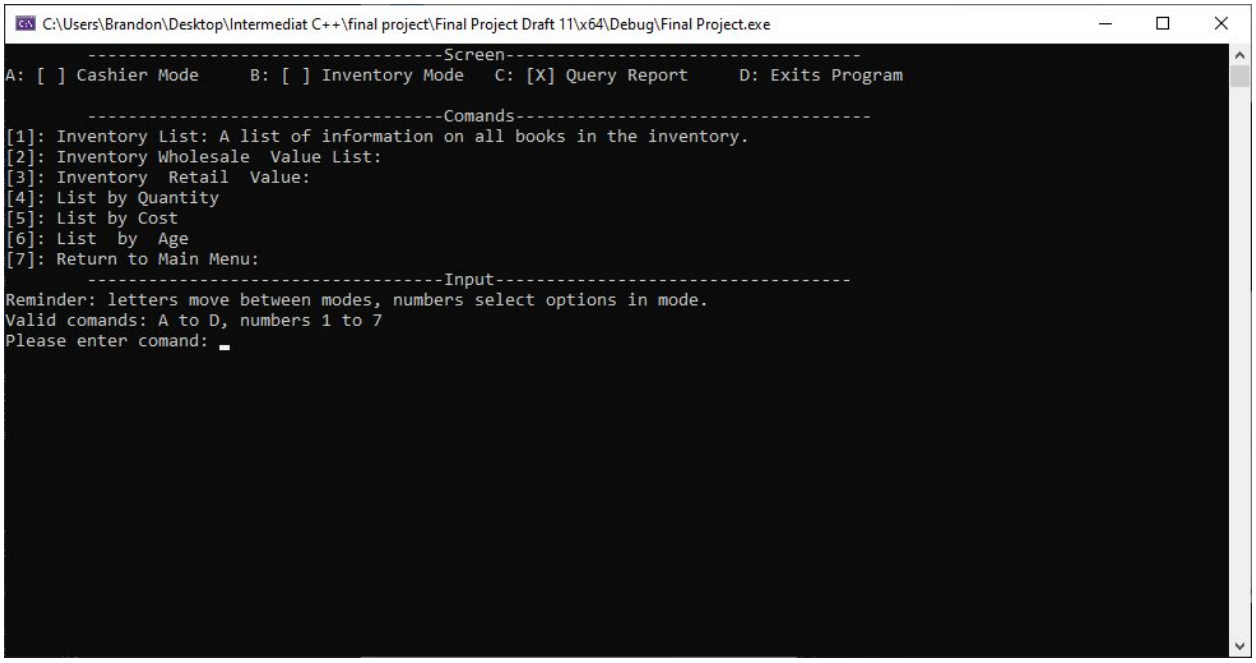


Invalid entries

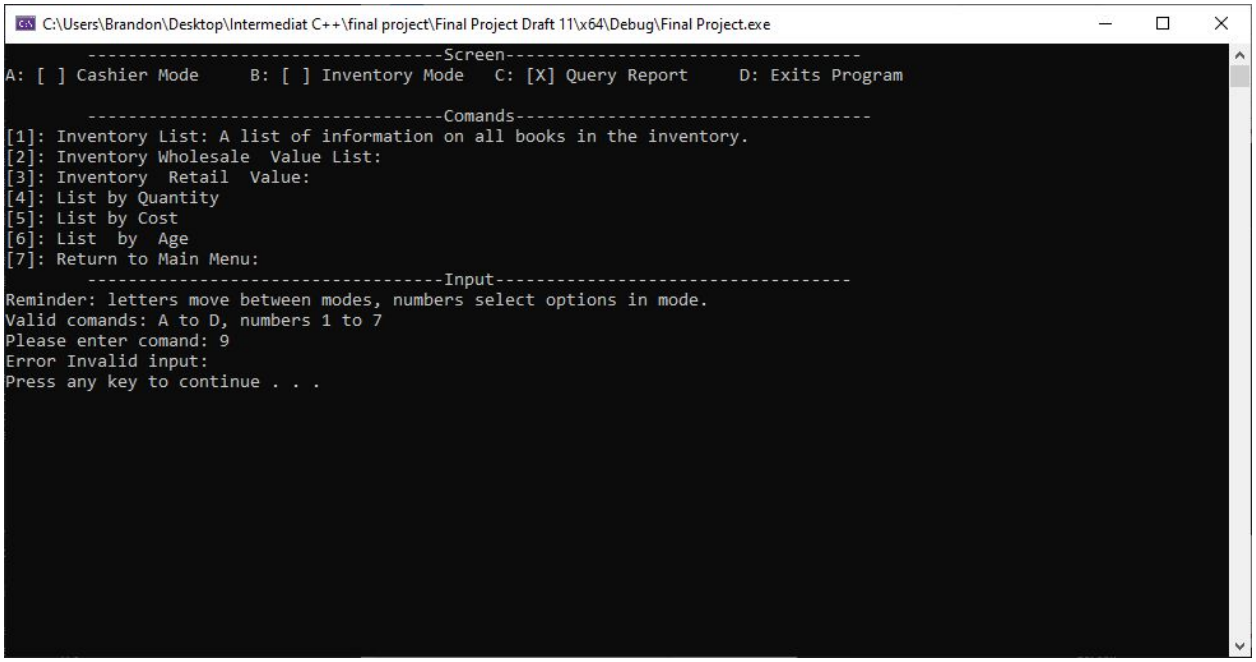


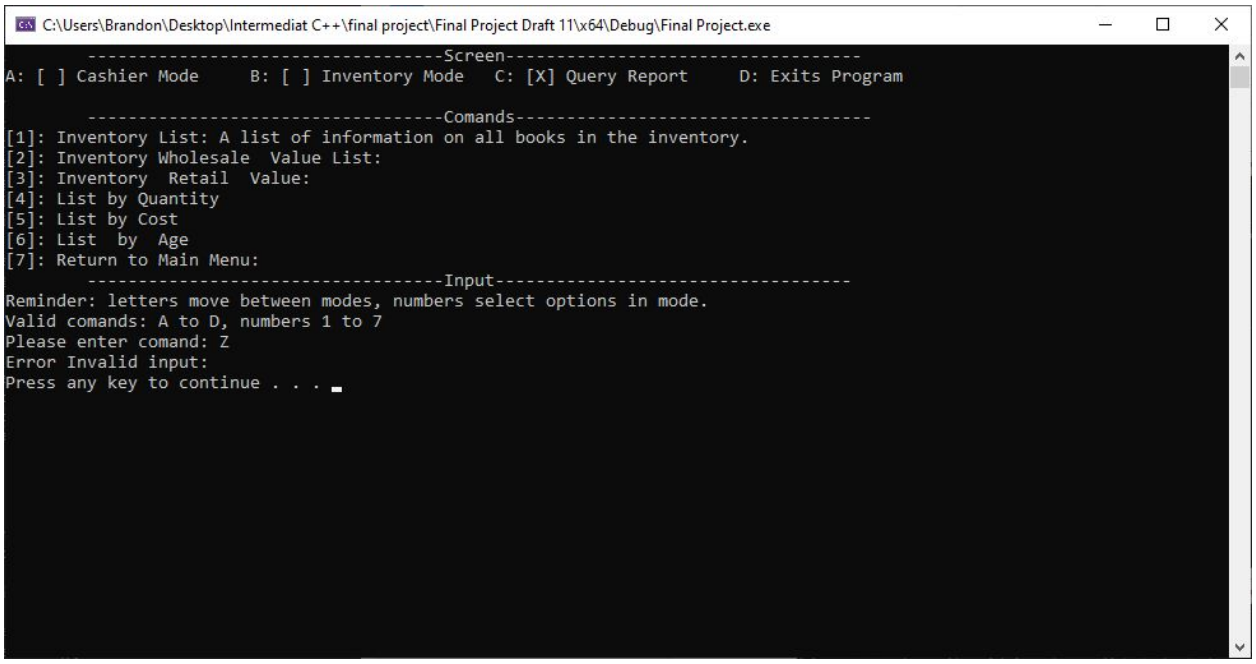


Report Module:

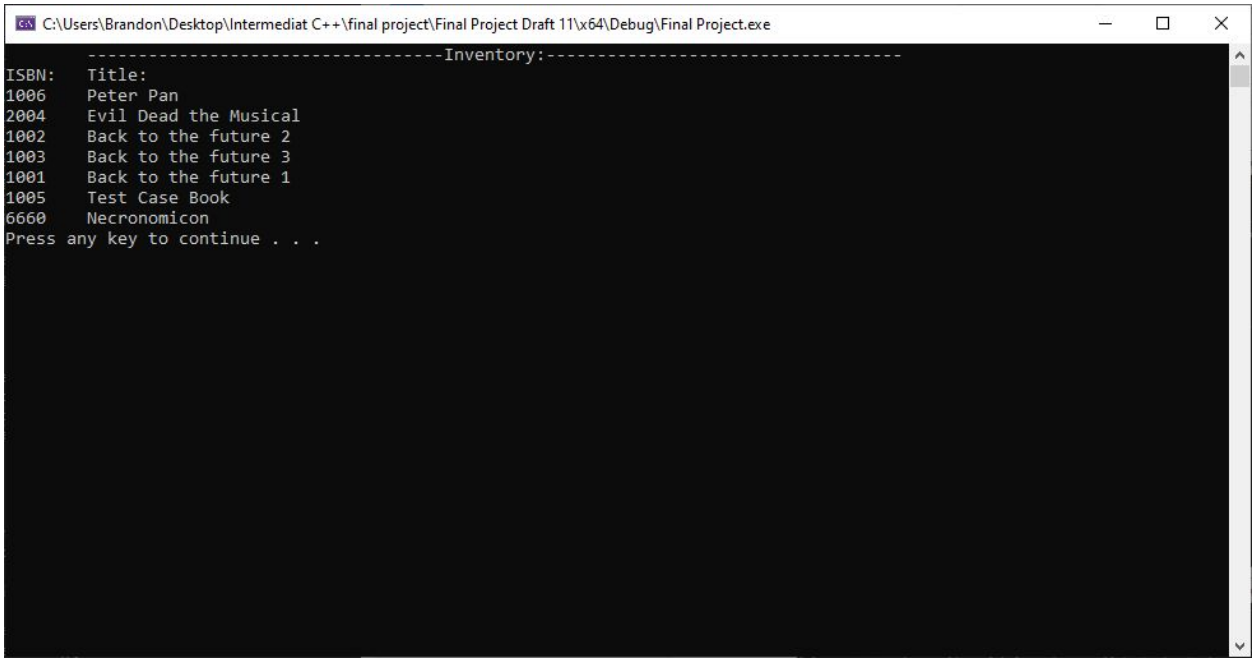


Invalid entries:

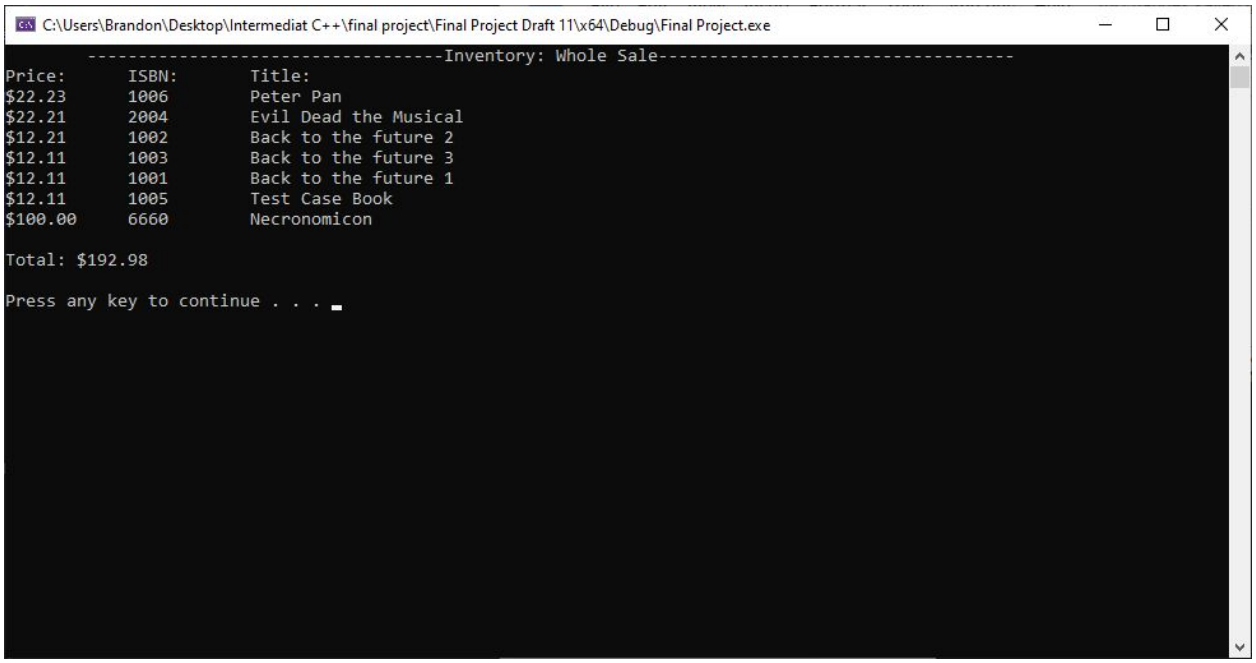




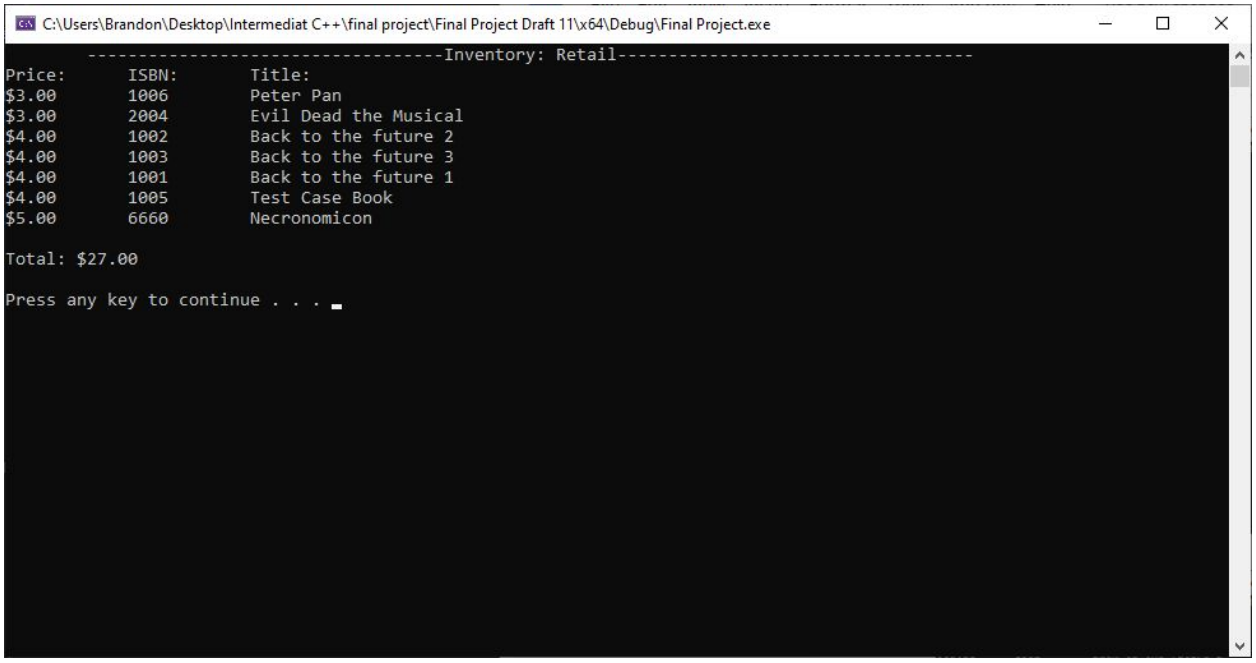
Report sub menu inventory list:



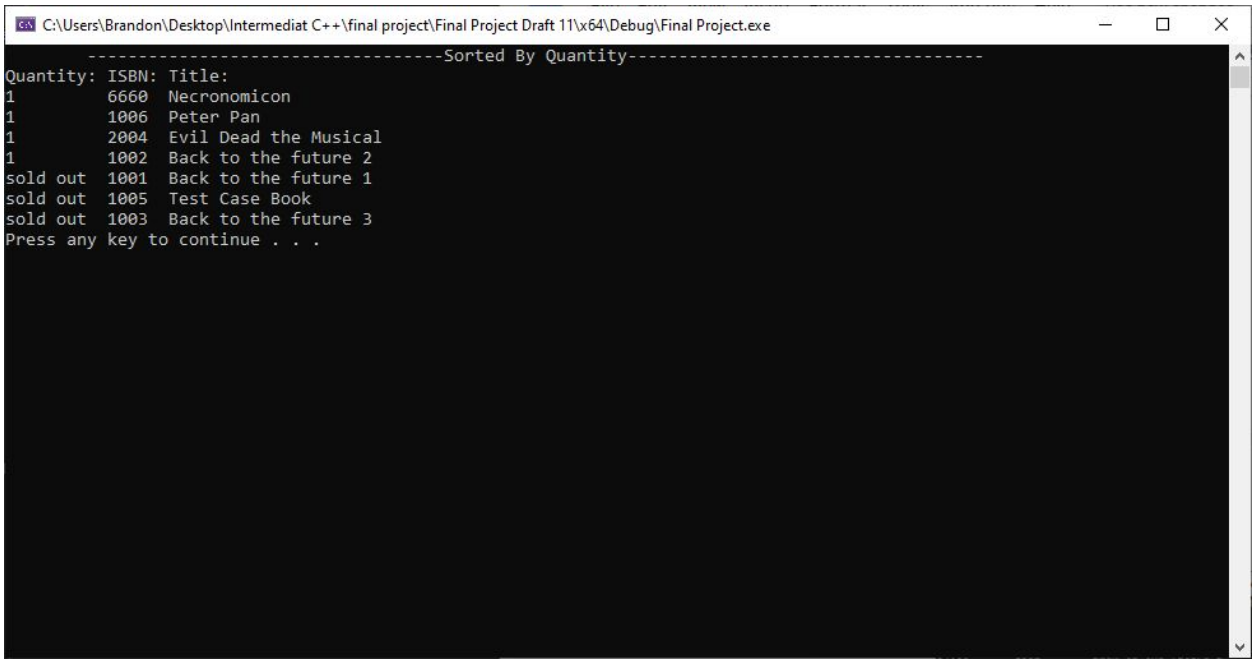
Report sub menu inventory whole sale list:



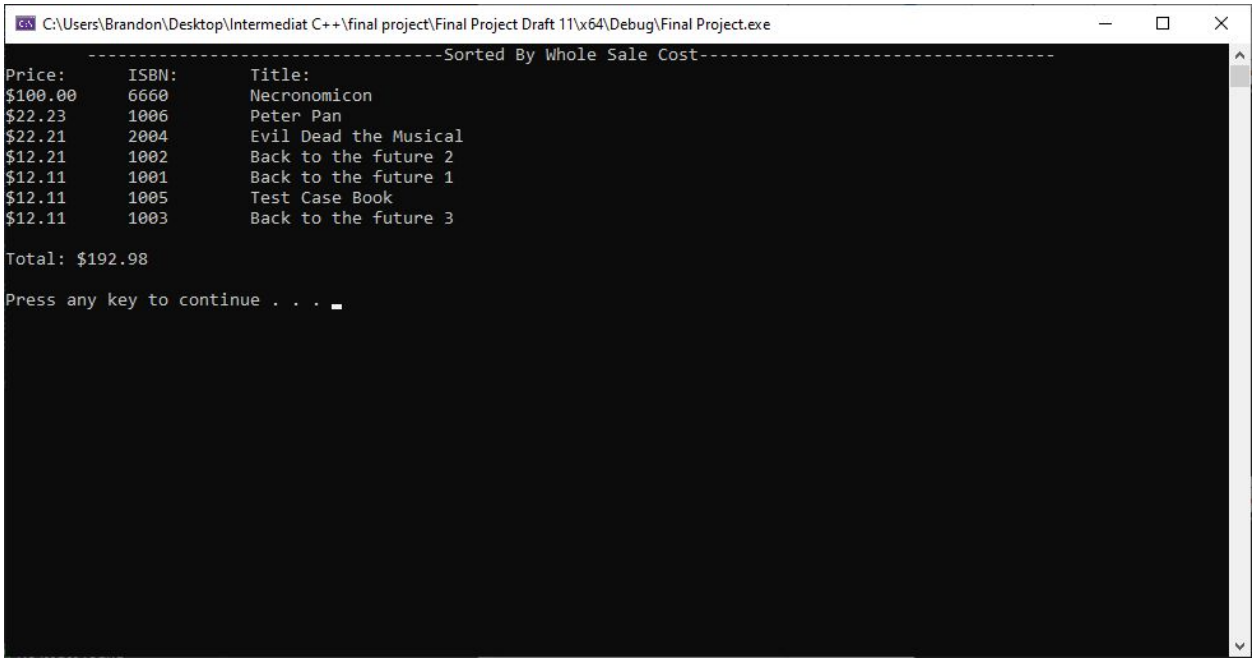
Report sub menu inventory Retail list:



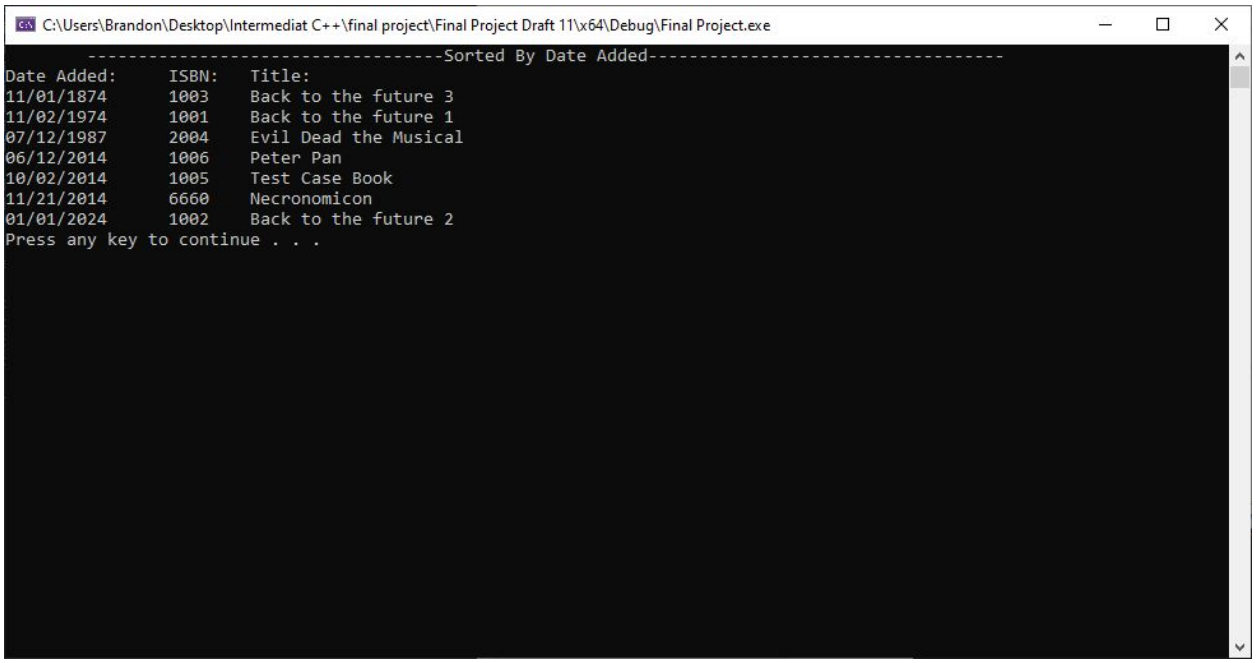
Report Sub Menu Sort by Quantity:



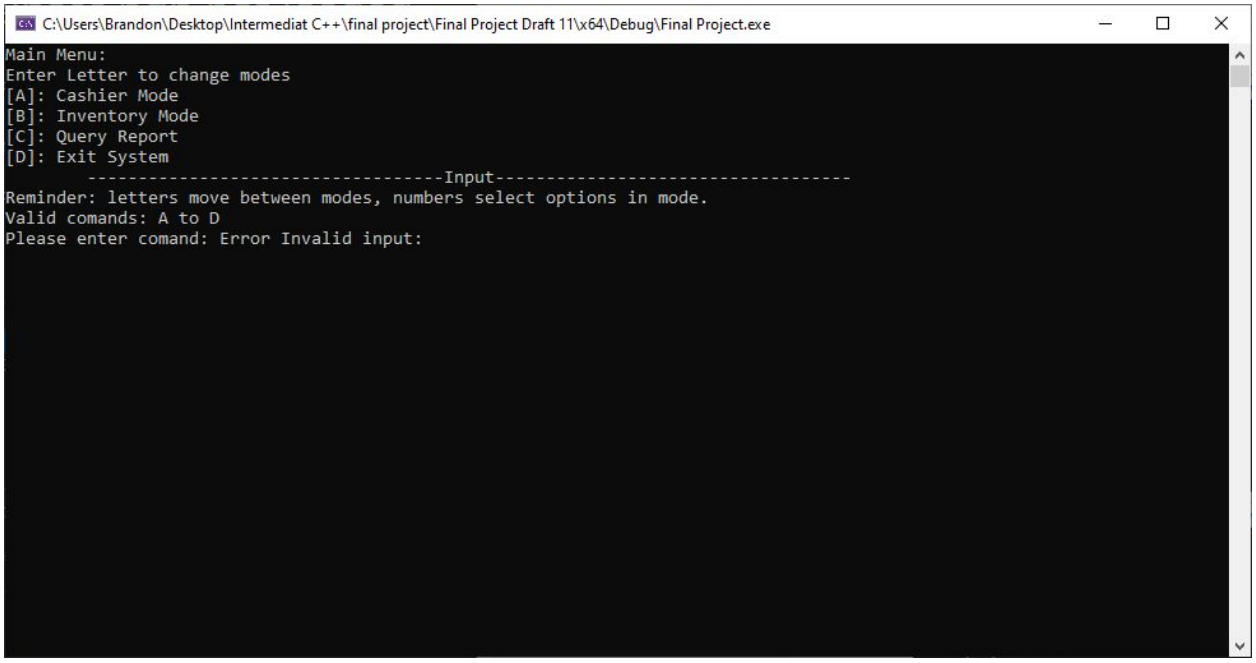
Report Sub Menu Sort by Whole Sale Cost:



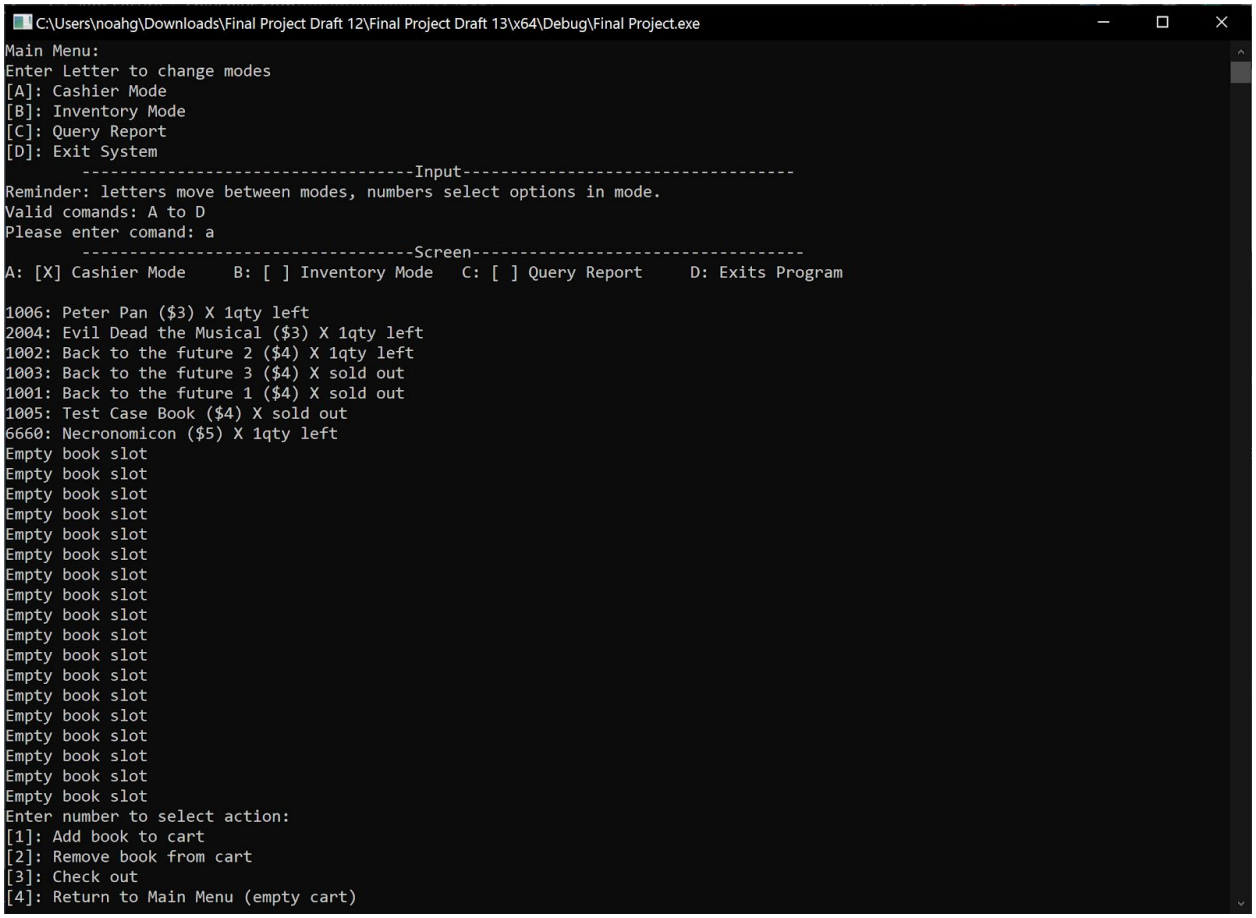
Report Sub Menu Sort by Date Added:



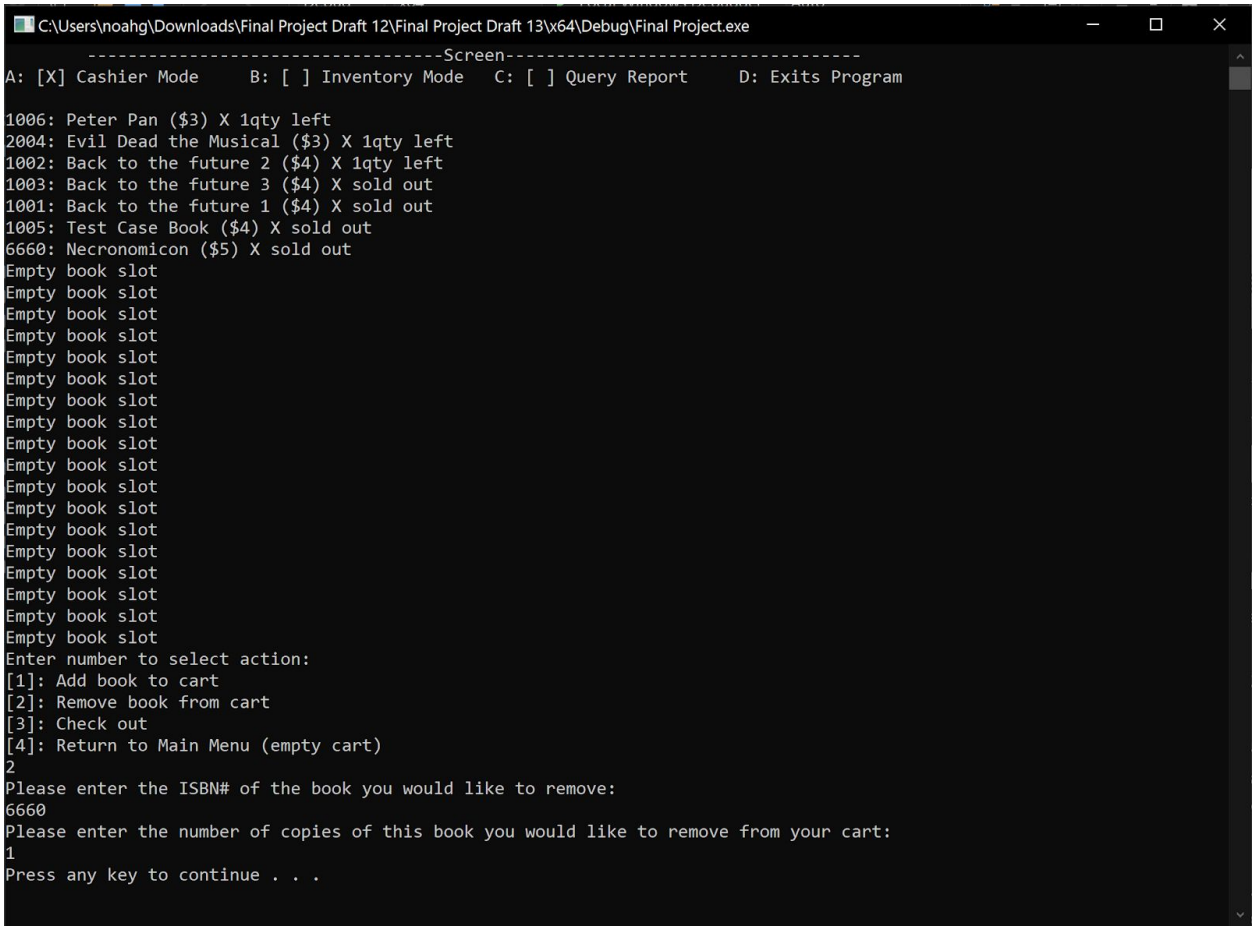
Exit Report Menu:



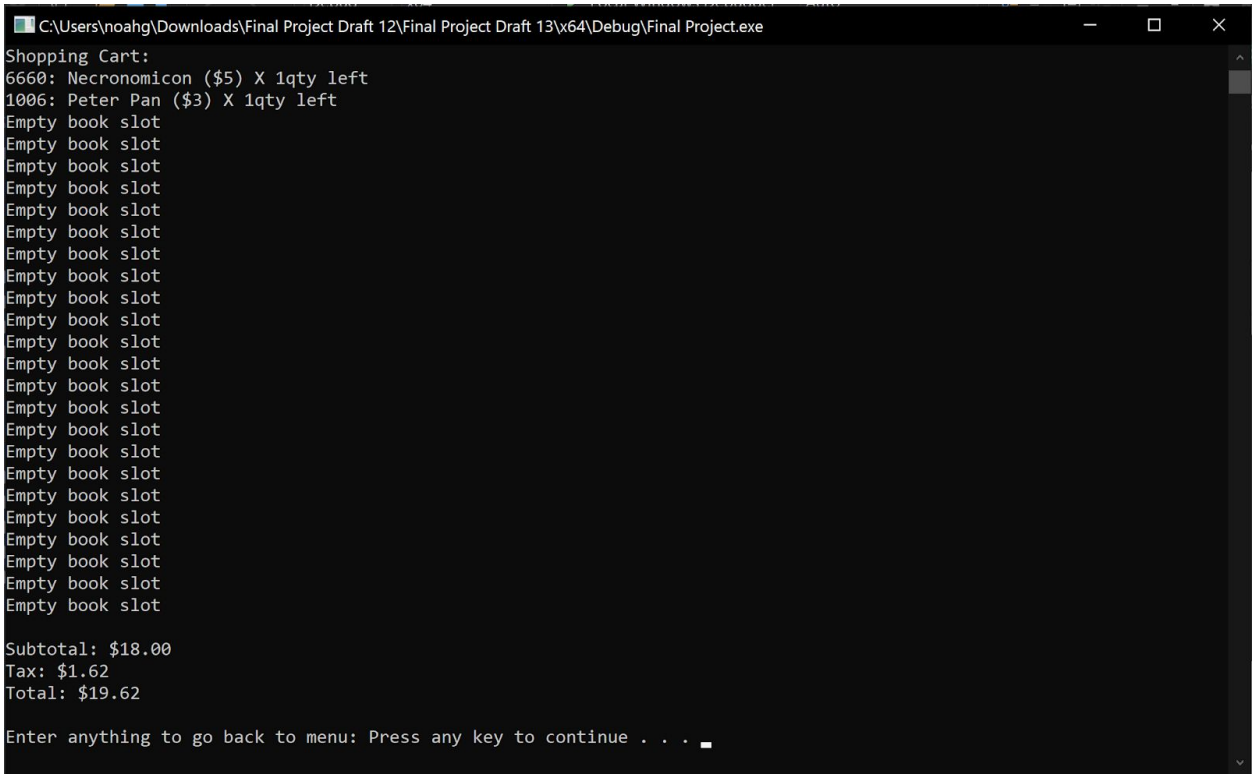
Cashier Mode:



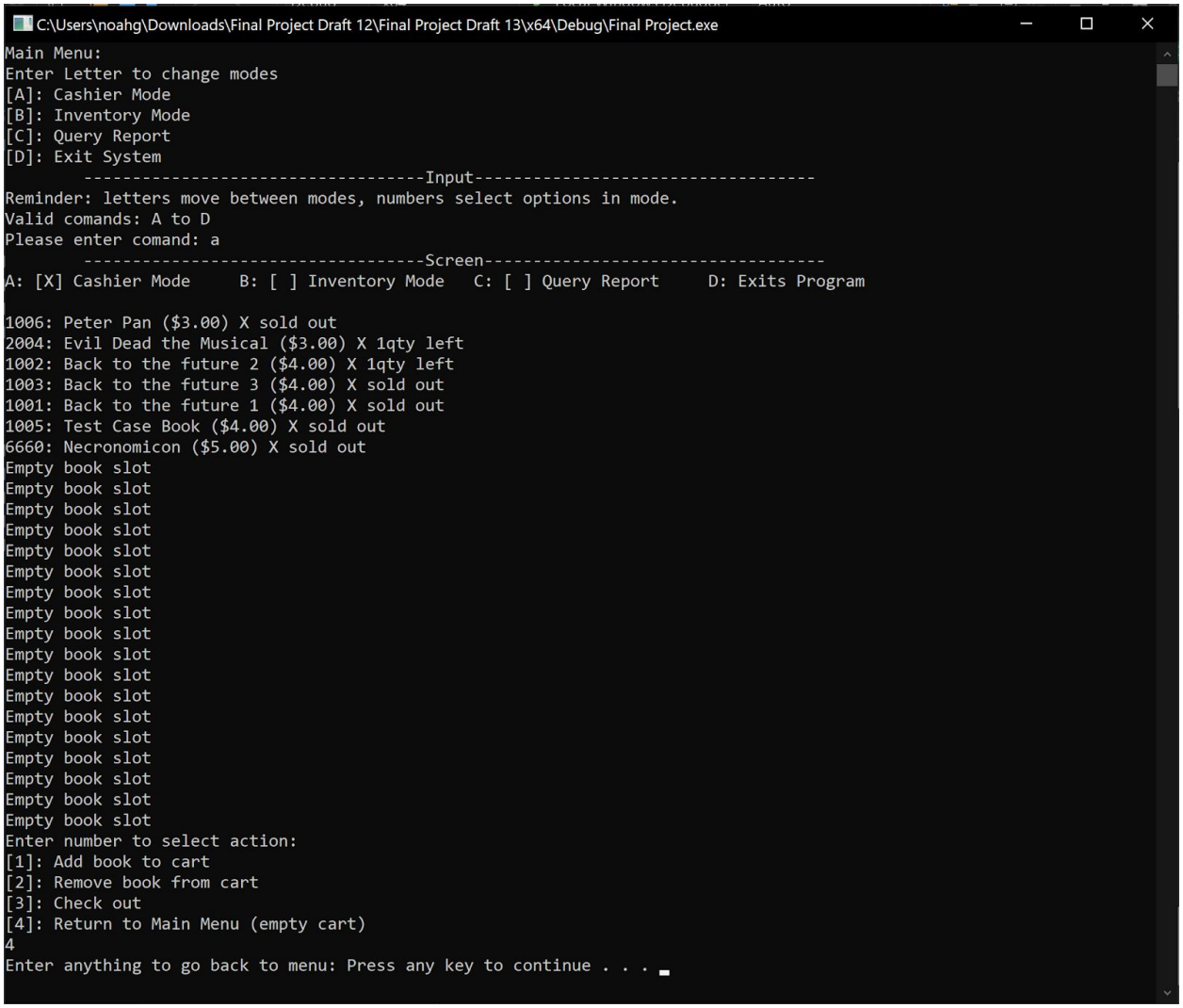
Removing Book from Cart:



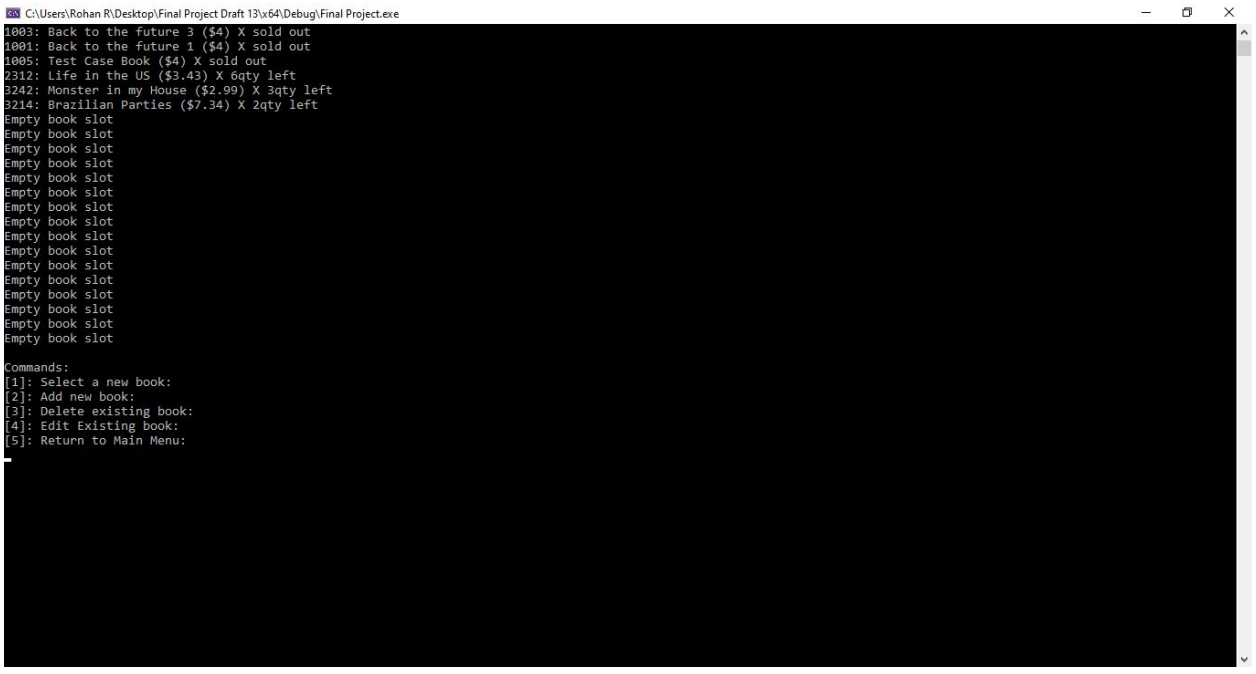
Check Out:


C:\Users\noahg\Downloads\Final Project Draft 12\Final Project Draft 13\x64\Debug\Final Project.exe
Shopping Cart:
6660: Necronomicon ($5) X 1qty left
1006: Peter Pan ($3) X 1qty left
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Empty book slot
Subtotal: $18.00
Tax: $1.62
Total: $19.62
Enter anything to go back to menu: Press any key to continue . . .

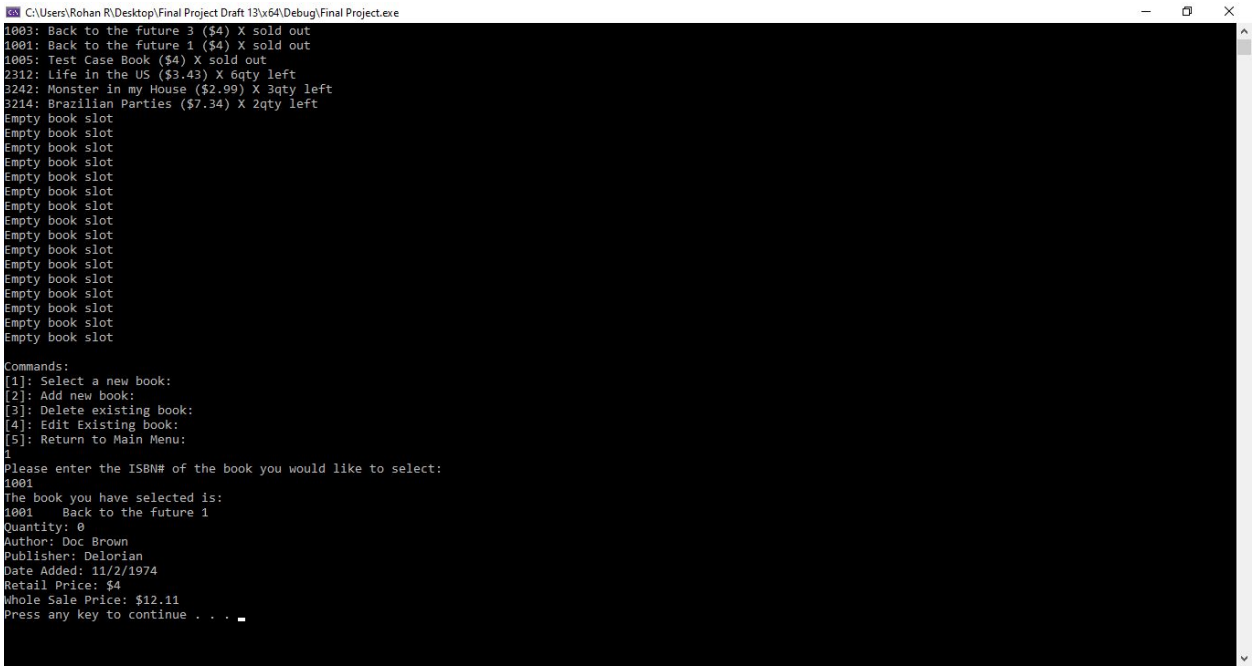
Returning to Main Menu:



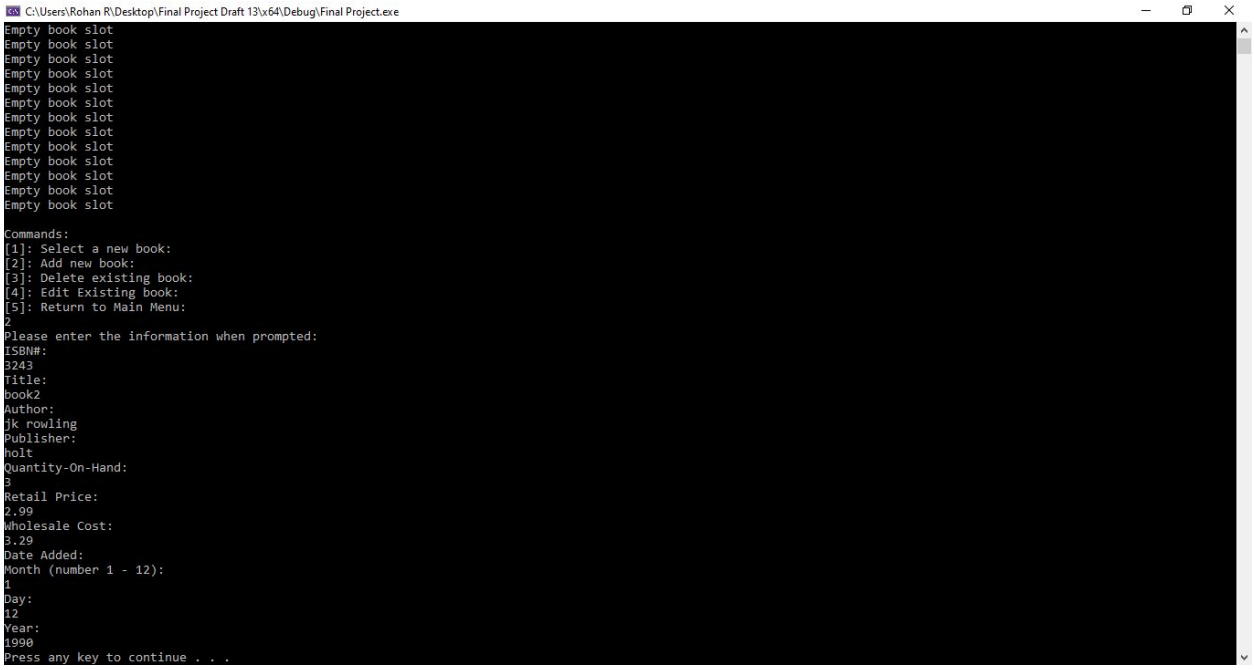
Inventory Module Main Menu:



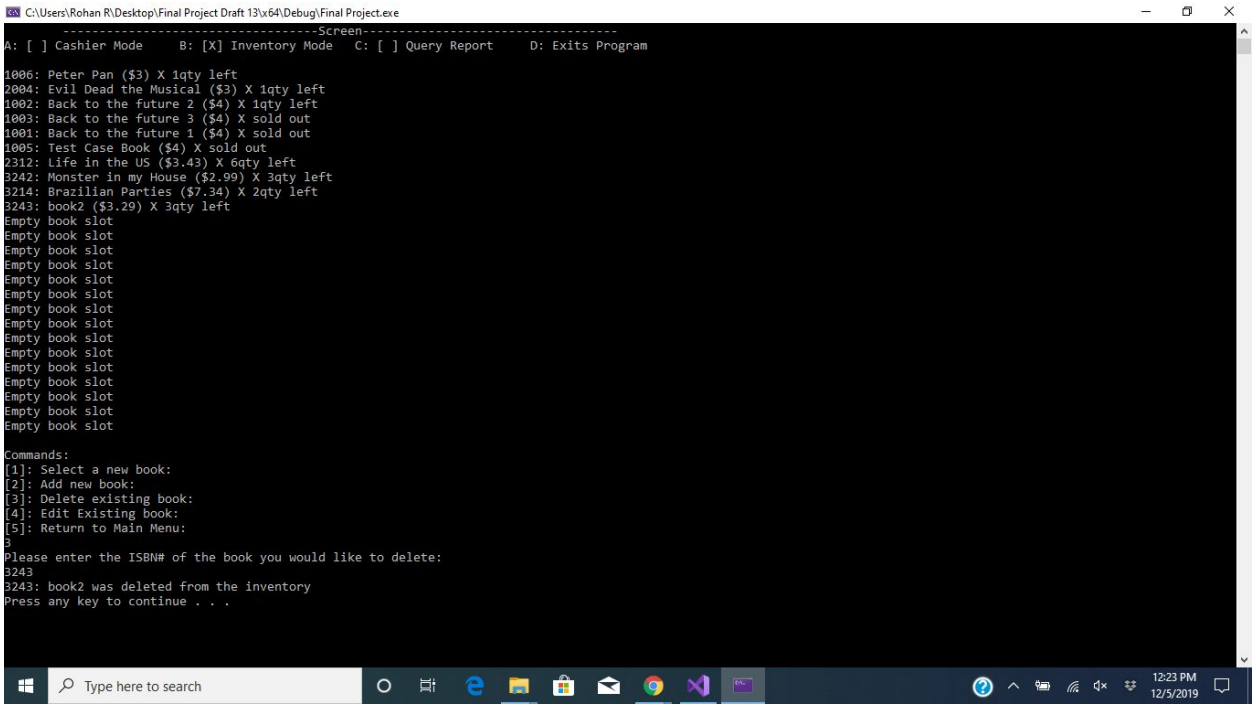
Command 1 Inventory Module:



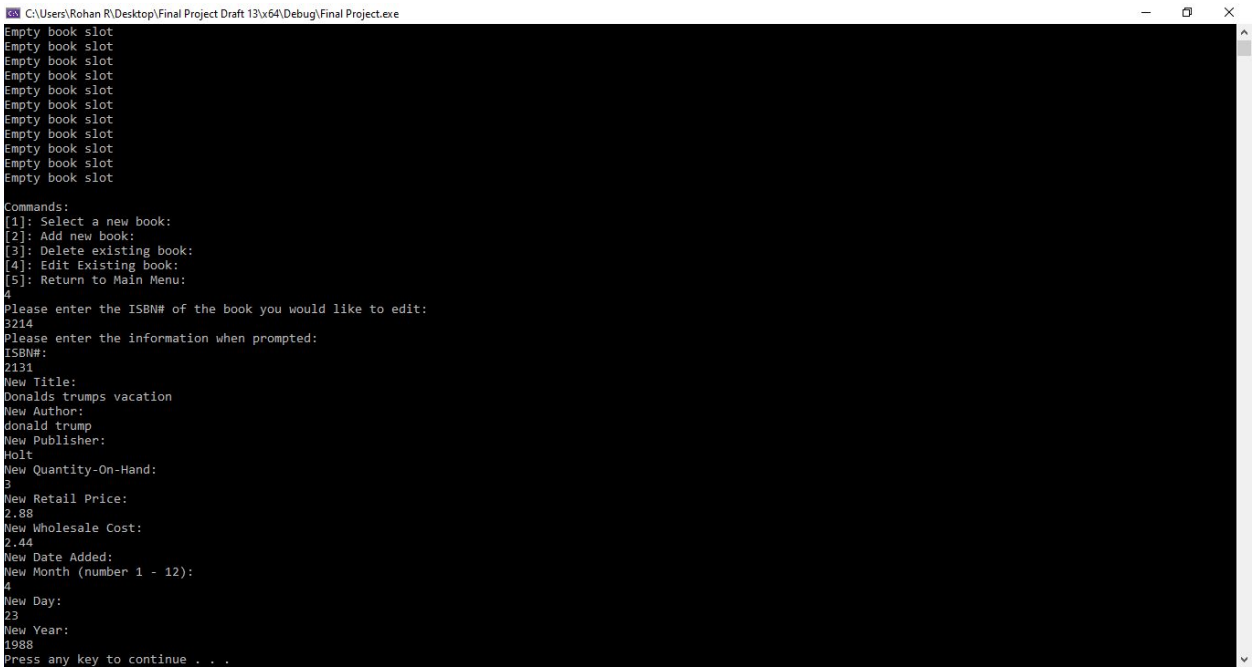
Command 2 Inventory Module:



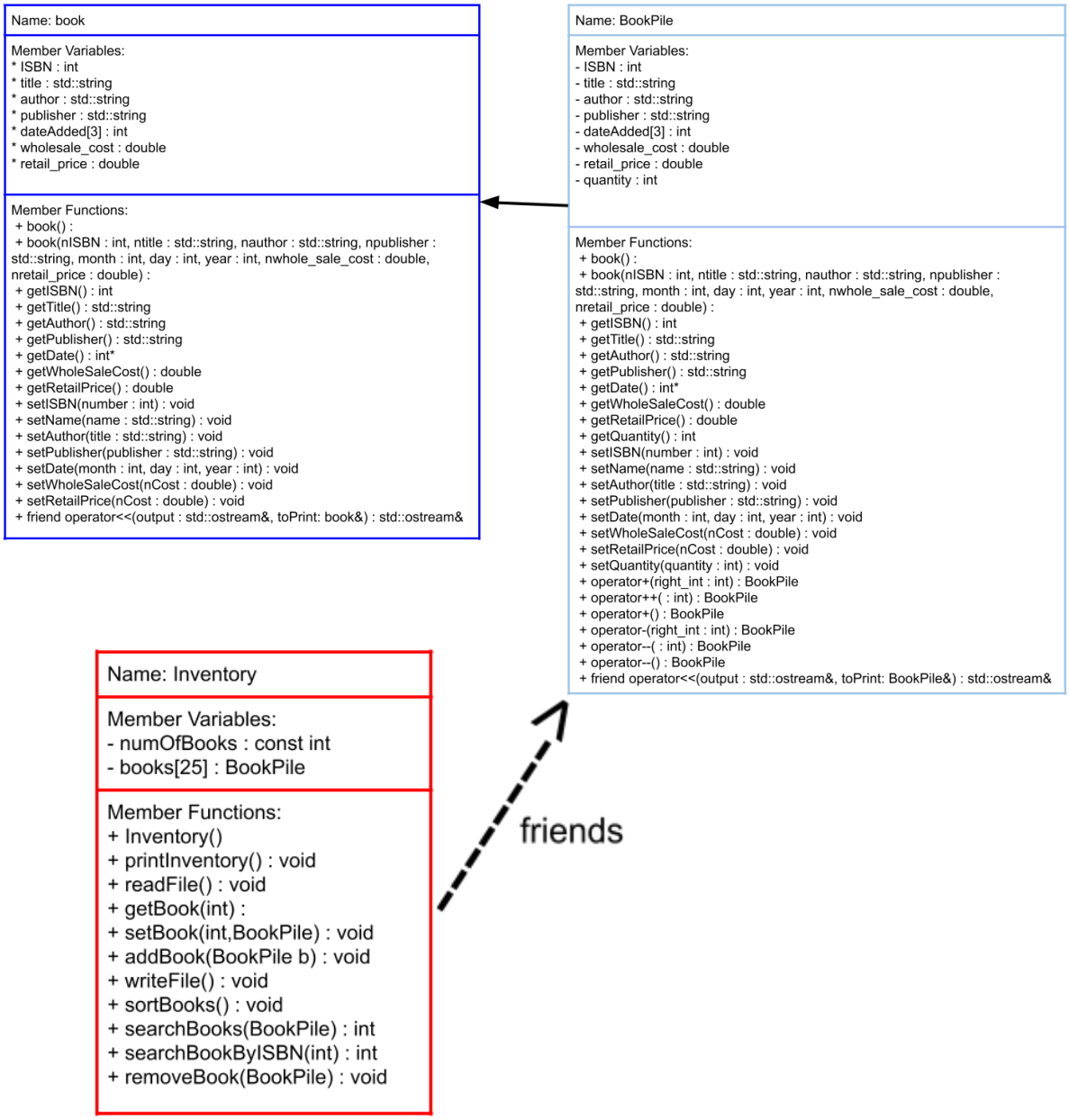
Command 3 Inventory Module:



Command 4 Inventory Module:



UML:


Name: book
Member Variables:
* ISBN : int
* title : std::string
* author : std::string
* publisher : std::string
* dateAdded[3] : int
* wholesale_cost : double
* retail_price : double
Member Functions:
+ book() :
+ book(nISBN : int, ntitle : std::string, nauthor : std::string, npublisher : std::string, month : int, day : int, year : int, nwhole_sale_cost : double, nretail_price : double) :
+ getISBN() : int
+ getTitle() : std::string
+ getAuthor() : std::string
+ getPublisher() : std::string
+ getDate() : int*
+ getWholeSaleCost() : double
+ getRetailPrice() : double
+ setISBN(number : int) : void
+ setName(name : std::string) : void
+ setAuthor(title : std::string) : void
+ setPublisher(publisher : std::string) : void
+ setDate(month : int, day : int, year : int) : void
+ setWholeSaleCost(nCost : double) : void
+ setRetailPrice(nCost : double) : void
+ friend operator<<(output : std::ostream&, toPrint: book&) : std::ostream&
Name: BookPile
Member Variables:
- ISBN : int
- title : std::string
- author : std::string
- publisher : std::string
- dateAdded[3] : int
- wholesale_cost : double
- retail_price : double
- quantity : int
Member Functions:
+ book() :
+ book(nISBN : int, ntitle : std::string, nauthor : std::string, npublisher : std::string, month : int, day : int, year : int, nwhole_sale_cost : double, nretail_price : double) :
+ getISBN() : int
+ getTitle() : std::string
+ getAuthor() : std::string
+ getPublisher() : std::string
+ getDate() : int*
+ getWholeSaleCost() : double
+ getRetailPrice() : double
+ getQuantity() : int
+ setISBN(number : int) : void
+ setName(name : std::string) : void
+ setAuthor(title : std::string) : void
+ setPublisher(publisher : std::string) : void
+ setDate(month : int, day : int, year : int) : void
+ setWholeSaleCost(nCost : double) : void
+ setRetailPrice(nCost : double) : void
+ setQuantity(quantity : int) : void
+ operator+(right_int : int) : BookPile
+ operator++(: int) : BookPile
+ operator+() : BookPile
+ operator-(right_int : int) : BookPile
+ operator--(: int) : BookPile
+ operator--() : BookPile
+ friend operator<<(output : std::ostream&, toPrint: BookPile&) : std::ostream&
Name: Inventory
Member Variables:
- numOfBooks : const int
- books[25] : BookPile
Member Functions:
+ Inventory()
+ printInventory() : void
+ readFile() : void
+ getBook(int) :
+ setBook(int,BookPile) : void
+ addBook(BookPile b) : void
+ writeFile() : void
+ sortBooks() : void
+ searchBooks(BookPile) : int
+ searchBookByISBN(int) : int
+ removeBook(BookPile) : void
friends