**struct product**

declare itemNum;

declare itemName;

declare itemCost;

declare itemQty;

declare struct product next pointer, initialize to NULL

declare struct product previous pointer, initialize to NULL

**struct cart**

declare itemNum;

declare itemQty;

declare struct cart next pointer, initialize to NULL

declare struct cart previous pointer, initialize to NULL

**Function main**

***Input:*** inFile, outFile, product\* phead, cart\* chead, adminOption, productNum, productQty

***Processing:*** lengthList, productExists, qtyExists, enterCart, adminOrBuyer, productFound, customerInfo[5], endAdmin, count

***Output:*** N/A

***Algorithm***

1. Open fileName
2. If filename opens

Call BuildProductList()

Initalize lengthList to length()

Display “If you are an admin enter admin. If you are a buyer enter buyer.”

Store output into adminOrBuyer

If adminOrBuyer = “admin”

Display "You are currently in admin mode!"

Do

Display “to add product enter add, to delete product enter delete, to update product qty enter update”

Store output into adminOption

Call adminOptions()

Display “exit admin mode enter exit”

Store output into endAdmin

While(endAdmin != “exit”)

Call updateInventory() function

End program

Else if adminOrBuyer = “buyer”

Display “If you would like to continue to checkout, enter 0.”

Display “If you would like to view your cart, enter cart.”

For (count=0, count < lengthList , count++)

Set productNum to productPurchase()

If (enterCart=true)

Check to see if the cart is empty

If (chead=NULL)

Display Cart is empty if it is

Else

Call PrintCartLinkedList()

If (productNum=0)

Display “name”

Get customerInfo[0]

Display “street address”

Get customerInfo[1]

Display “city”

Get customerInfo[2]

Display “state”

Get customerInfo[3]

Display “zip code”

Get customerInfo[4]

Call displayInvoice()

Call updateInventory()

Call deleteList()

Break loop

Else

Set productExists to searchProduct()

If (productExists=false)

Display “product does not exist”

Else

Do

Display “What is the qty wanting to be purchased”

Store output into productQty

Set qtyExists to searchQty()

If (qtyExists=false)

Display “the quantity entered is not available”

While(qtyExists=false)

If(qtyExists=true)

Display “product has been added to your shopping cart”

Call BuildCartList()

Update lenghtList

**Function BuildProductList:** reads file from productData.csv and builds a linked list from the file

***Parameters:*** inFile by reference, product\* headRef by reference

***Processing:*** tempProduct, titemNum, titemName, titemCost, titemQty, titemNum1, titemName1, titemQty1, titemCost1

***Output:*** N/A

***Return type:*** void

***Algorithm***

1. Open inFile
2. If inFile is unable to open

End function

1. If file is able to open

Declare struct product\* head and initialize to NULL

Declare struct product\* current and initialize to NULL

While (end of File has not been reached)

Declare struct product\* newNode and initialize to dynamic product

Use getLine to get the itemNum

Set newNode itemNum to titemNum

Use getLine to get the itemProduct

Set newNode itemProduct to titemProduct

Use getLine to get the itemNum

Set newNode to itemCost to titemCost

Use getLine to get the itemNum

Set newNode to itemQty to titemQty

If (head=NULL)

Set head to newNode

Set current to head

Else

Set current next to newNode

Set newNode previous to current

Set current to newNode

1. Close file
2. Set headRef to head

**Function length:** finds the length of the product linked list

***Parameters****:* product\* head

***Processing:*** count

***Output:*** count

***Return type:*** int

**Algorithm**

1. Declare struct product\* current and initialize it to product\* head
2. Declare count and initialize to 0
3. While (current !=NULL)

Count++

Set current to current next

1. Return count

**Function productPurchase:** asks the user which product they would like to purchase and return the product number

***Parameters****:* cart by reference

***Processing:*** productNumber, answer

***Output:*** productNumber

***Return type:*** int

**Algorithm**

1. Display “What product would you like to purchase”

Store output in answer

1. If (answer=”cart”)

Store cart as true

Else

Store cart as false

Convert answer to an int and store it in productNumber

1. Return productNumber

**Function searchProduct:** searchs for the product number in the product linked list

***Parameters:*** product\* headRef by reference, productNum, productFound by reference

***Processing:*** count

***Output:*** productExists

***Return type:*** Bool

***Algorithm***

1. Declare struct product\* current and initialize it to head
2. Declare count and initialize it to 0
3. While (current != NULL)

If (current itemNum = productNum)

Output product Number and product Cost

Set productFound to count

Return true

Else

Set current to current next

Count++

1. Return false

**Function searchQty:** checks if the quantity being purchased is available

***Parameters:*** product\* headRef by reference, productNum, productFound by reference, productQty

***Processing:*** N/A

***Output:*** N/A

***Return type:*** Bool

***Algorithm***

1. Declare struct product\* current and initialize it to head
2. While (current!=NULL)

If(current itemNum =productNum)

If (current itemQty > productQty

Set current itemQty to current itemQty-productQty

Return true

Return false

Else

Set current to current next

1. Return false

**Function BuildCartList:** builds the cart linked list if the product and quanity wanted exists

***Parameters:*** cart\* headRef by reference, productNum, itemQty

***Processing:*** productPurchased, productFound, count

***Output:*** n/a

***Return type***: void

***Algorithm:***

1. ***Declare struct cart*** Declare struct cart\* head and initialize it to headRefCart
2. Declare struct cart current and initialize it to NULL
3. Declare struct cart newNode and initialize it to dynamic memory cart
4. Set newNode itemNum to productNum
5. Set newNode itemQty to itemQty
6. Set newNode next to NULL
7. If (head does not exist)

Head is set to newNode

Else

While (Current !=NULL)

If ( current itemNum = product Num)

productPurchased set to true

break while loop

Else

Set current to current next

Set productPurchased to false

If(productPurchased=true)

Set current to head

Update current itemNum with newQty+oldQty

Else

Set current node to head node

Traverse the linked list until you reach the end

At end set current to point to new node

1. Set headRefCart to head node

**Function printCartLinkedList:** displays the current cart linked list

***Parameters:*** struct cart\* head, struct product\* phead

**Processing:** total

***Output:*** N/A

***Return type:*** void

***Algorithm***

1. Declare struct product\* current and set it to phead node
2. Declare struct cart\* current and set it to head node
3. Declare total and set it to 0.0
4. Display “item number, item name, qty, unit cost, total”
5. While (currentp != NULL)

While (currentc !=NULL)

If (currentp itemNum = currentc itemNum)

Update total

Output the item number, item name, itemQty being purchased, current itemTotal

Set currentc node to currentc next

Set currentp node to currentp next

**Function displayIvoice:** displays the invoice to the console and create invoice.txt

***Parameters:*** customerInfo[5], cart\* head, product\* phead

**Processing:** total

***Output:*** N/A

***Return type:*** void

***Algorithm***

1. Open outFile “Invoice.txt”
2. If ofstream does not open

Return 0

Else

Declare struct product\* current and set it to phead node

Declare struct cart\* current and set it to head node

Declare total and set it to 0.0

Display customerInfo to console

Output customerInfo to outFile

Display “item number, item name, qty, unit cost, total” to console

Outuput “item number, item name, qty, unit cost, total” to outFile

While (currentp != NULL)

While (currentc !=NULL)

If (currentp itemNum = currentc itemNum)

Update total

Output the item number, item name, itemQty being purchased, current itemTotal

Set currentc node to currentc next

Set currentp node to currentp next

Display final total value to console

Output final total value to outFile

**Function updateInventory:** updates the ProductData.csv file with the inventory remaining after products have been purchased

***Parameters:*** outfile by reference, product\* head

**Processing:** product\* current

***Output:*** n/a

***Return type:*** void

***Algorithm***

1. Opens productData.csv
2. If unable to open outputs error
3. Else

While (current != NULL)

If (current itemQty = 0)

Set current node to current next

Else

Output the current itemNum to outFile

Output the current itemName to outFile

Output the current itemCost to outFile

Output the current itemQty to outFile

Set current node to current next

Close the outFile

**Function DeleteList:** deletes the product and cart linked list

***Parameters:*** product\* head by Reference, cart\* chead by Reference

**Processing:** product\* pcurrentNode, product\* pnextNode, cart\* ccurrentNode, cart\* ccurrentNode

***Output:*** n/a

***Return type:*** void

***Algorithm***

1. Set pcurrentNode to product headRef
2. Set pnextNode to product headRef
3. Set ccurrentNode to cart headRef
4. Set cnextNode to cart headRef
5. While (pcurrentNode != NULL)

Set the next node

Delete the current node

Set current node to the next node

Set pheadRef to nullptr

1. While (ccurrentNode != NULL)

Set the next node

Delete the current node

Set current node to the next node

Set cheadRef to nullptr

**Function adminOptions:** if the admin enters admin status they are given the option to add product, delete product, update productsQty

***Parameters:*** adminOption, product\* head by reference, productFound by reference

**Processing:** nitemNum, nitemName, nitemQty, nitemCost, ditemNum, uitemNum, uitemQty

***Output:*** n/a

***Return type:*** void

***Algorithm***

1. If (adminOption = “add”)

Ask the user for the new product number, cost of the new product, and qty of new product

Store output in nitemNum, nitemName, nitemQty

Call insertProduct()

1. If(adminOption = “delete”)

Ask the user “what is the product number the would like to delete”

Store ouput in ditemNum

If (searchProduct()=true)

Call deleteProduct()

Else

Display “the product does not exist”

1. If(adminOption = “update”)

Ask the user “what is the product number the would like to update”

Store ouput in uitemNum

If (searchProduct()=true)

Ask the user “new qty of the product”

Call insertQty()

Else

Display “the product does not exist”

**Function insertProduct:** updates the ProductData.csv file with the inventory remaining after products have been purchased

***Parameters:*** product\* head by reference, ntemNum, nitemName, nitemCost, nitemQty

**Processing:** product\* head, product\* current, product\* newNode

***Output:*** n/a

***Return type:*** void

***Algorithm***

1. Declare product\* head and set it to head node
2. Declare product\* current and set it to NULL
3. Declare product\* newNode and set it to dynamic product memory
4. Set newNode itemNum, itemName, itemCost, itemQty to ntemNum, nitemName, nitemCost, nitemQty
5. Set newNode next to NULL
6. If (head does not exist)

Head node is set to newNode

Else

Current is set to head node

While (current -> next)

Current node is set current next

Current next is set to newNode

Shet headRef to head node

**Function deleteProduct:** updates the ProductData.csv file with the inventory remaining after products have been purchased

***Parameters:*** product\* head by reference, position

**Processing:** product\* temp, count

***Output:*** n/a

***Return type:*** void

***Algorithm***

1. If (head = NULL)

Return to main

1. Declare struct product\* temp and set it to head node
2. If (position=0)

Set head node to temp next

Free temp node

Return to main

1. Declare count and initialize it to 0
2. While ( temp!=NULL and count < position--)

Set temp node to temp next

1. If (temp =NULL or temp-> next=NULL)

Return to main

1. Declare struct product\* next and initialize it to temp next next
2. Free temp next
3. Set temp next to next node

**Function insertQty:** updates the ProductData.csv file with the inventory remaining after products have been purchased

***Parameters:*** product\* head by reference, productNum, productQty

**Processing:** product\* current

***Output:*** n/a

***Return type:*** void

***Algorithm***

1. Declare struct product\* current and set it to head node
2. While (current !=NULL)

If (current itemNum = productNum)

Set current itemQty to productQty

Break

Else

Set current node to current next

1. Set head node to current