~~1.~~ **~~Specification of requirements [25]~~**

* Definition of problem **(5)** - Complete accurate description of the problem 5
* Techniques of analysis used **(5)** - Sound and relevant techniques used 5
* Use of Data Flow diagrams and E-R diagrams **(9)**
* Context Level **(3)** - Complete and accurate diagram of all relevant entities, data flows
* Level 1 Diagram **(3)** - Complete and accurate diagram of all relevant processes, data flows and major data stores
* Entity Relation Diagram (ERD) **(3)** - Complete and accurate diagram of all relevant entities and relationships.
* Functional and non-functional requirements **(6)**
* Functional Requirements **(3)**
* Non-Functional Requirements **(3)**

2. **Design Specification [14]**

* + **System structuring (4)** - Complete and accurate diagram of all processes
  + User interface design **(2)** - Thorough analysis and appropriate justification of interfacedesign
  + Report design **(2)** - Appropriate and well implemented
  + Algorithm design **(3)**- Appropriate and well implemented algorithm design
  + Choice of appropriate data structures **(3)** - Appropriate and well implemented

3. **Coding and Testing [15]**

* + Code achieves functionality **(5)** - Code achieved functionality (documentation, error trapping, correct output, usability and reporting)
  + Code corresponds to design **(5)** - Code achieves all the design specifications
  + Test plans **(5)** - Test Plan with exhaustive data set



Year : 2024

Subject : Computer Science (Unit 2)

Candidate Numbers : 1600261113

1600261229

1600260443

160026

Candidates’ Name : Eliana Dookhoo

Khadine Rambert

Mikayla S. Roberts

Sharona Lochan

Title : Shoewear Corner Software

Teacher : Mrs. Gopee-Lochan

Table of Contents

[Specification of Requirements 4](#_Toc687048887)

[Problem Definition 5](#_Toc541913890)

[Techniques of Analysis 5](#_Toc1502140669)

[Data Flow Diagram Context Level 7](#_Toc847098071)

[Data Flow Diagram Level 1 8](#_Toc100640358)

[Entity Relationship Diagram 9](#_Toc82432375)

[Functional Requirements 10](#_Toc254005011)

[Non-Functional Requirements 11](#_Toc775032783)

[Design Specification 12](#_Toc1251921255)

[System Structuring 13](#_Toc1614623354)

[User Interface Design 13](#_Toc170598792)

[Report Design 14](#_Toc1382018526)

[Algorithm Design 15](#_Toc359067267)

[Data Structures 59](#_Toc1324776051)

[Variable Table 61](#_Toc1186674969)

[Coding and Testing 63](#_Toc451160426)

[Code 64](#_Toc1936902606)

[Test Plan 110](#_Toc1100690025)

# Specification of Requirements

## Problem Definition

Shoewear Corner is a shoe store located on High Street, San Fernando that sells shoes to its customers through a membership system.

Customers come to the business’ physical location to purchase shoes. In order to do so, they have to become a member through a sign-up process. They are asked to provide particular information on a paper-based form, such as their name, age, birthday, gender, address and email. This is written in a book for record keeping. Subsequently, the information on the customer is stored in a file that is referenced whenever the customer makes a purchase at the business. However, when changes to customer information needs to be edited or revised, it is difficult to update the file. Instead, an employee would have to recreate a form and add it to the current existing member file.

Additionally, as the business’ clientele base grew, it became increasingly difficult to easily manage and access member information. It takes a long period of time for employees to access files due to the files being stored in cabinets in alphabetical order. Upkeep of this physical method of organization tends to be time-consuming and challenging for employees to accurately update and maintain the system of order. Moreover, customer’s purchasing information and balance details are stored in their member files which have to be retrieved during real time purchases and has proven to be an arduous task when updating members’ financial records.

difficulties keeping stock,

## Techniques of Analysis

**Observations**

Week 1 – February 1, 2023 to February 7, 2023:

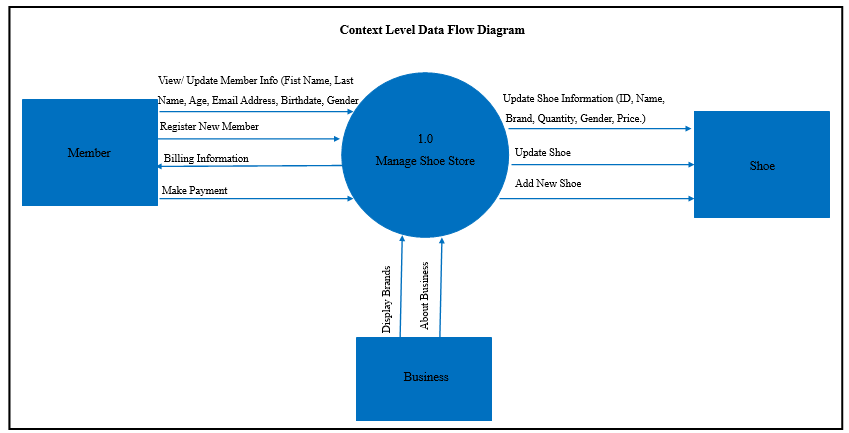
During the first week of observation, several challenges were observed at Shoewear Corner. The store experienced stock outs of popular shoe sizes, leading to lost sales opportunities and customer dissatisfaction. Additionally, there were instances of overstocking on less popular shoe sizes, tying up capital and shelf space. The store struggled to keep accurate records of customer purchases and membership information, leading to delays in processing returns and exchanges. This resulted in frustrated customers and a negative impact on the store's reputation. Customers reported long wait times at the checkout counter due to manual inventory checks and record-keeping processes. This led to a decline in customer satisfaction and a decrease in customers revisiting the store.

Week 2 – February 8, 2023 to February 14, 2023:

During the second week of observation, the challenges at Shoewear Corner persisted. The store continued to experience stock outs and overstocking, leading to loss of sales opportunities and inefficient use of resources. The store's record-keeping processes remained manual, resulting in delays in processing returns and exchanges. Customers continued to report long wait times at the checkout counter, impacting their overall shopping experience.

Based on the observations made over the two-week period, it was evident that Shoewear Corner was facing significant challenges in inventory management, record-keeping, and customer experience. Implementing technology solutions such as an inventory management system can help address these challenges, improve efficiency, and enhance the overall customer experience. It is imperative that the owner, Raquel, considers the benefits of technology implementation and takes steps to modernize the store's operations. Failure to do so may result in continued inefficiencies and a decline in customer satisfaction.

## Data Flow Diagram Context Level



## Data Flow Diagram Level 1

## 

## Entity Relationship Diagram

## Functional Requirements

The system should be able to:

* Display main menu with options for Member, Shoe, Business and Exit
* Register a new member by asking user for necessary personal information
* View member information by displaying the personal information an existing member
* Give members their billing information
* Allow members to make a payment after purchasing from the business
* Add information for a new shoe
* Update the information of an existing shoe
* Display information about the business
* Display brands that are sold at the shoe store
* Return to main menu from any option
* Exit the program

## Non-Functional Requirements

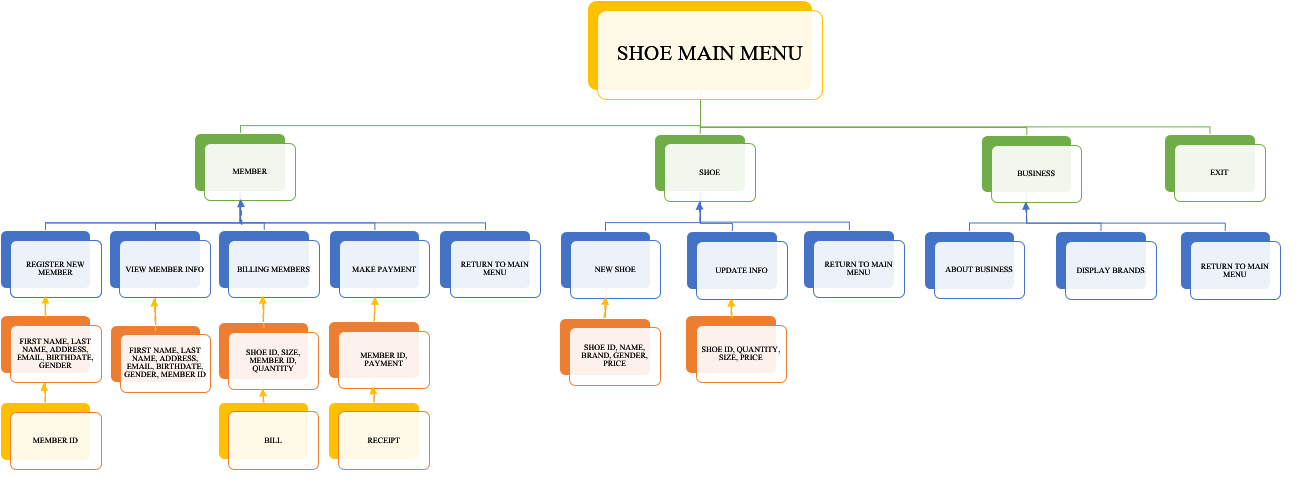
The system should be able to:

* Maintenance – the program must be able to make changes for future requirements
* Efficiency – system should possess a high response time and utilize the memory of the
* Usability – the interface should be easy for all users to operate
* Portability – in the event that more devices are added to the business, the program should be able to be used by those devices
* Dependability – the program should be safe, secure and reliable

# Design Specification

## 

## System Structuring



## User Interface Design

The user interface intended to be used for this shoe store programme is a command driven interface. It will demonstrate the approach used to meet the needs of both the customer and the business.

The main menu page will provide clear options for different functionalities such as managing members, handling billing, managing shoe options, accessing shoe information and exiting the program. This layout will ensure easy navigation for the user which will allow them to easily access the desired features of this system.

The member management option will allow the system to store and retrieve member details such as member ID, full name, email, phone number, birthdate, gender and address. It will also allow for registration of new members, updating or viewing member information, billing members, making payments and returning to the main menu. This feature will be important for maintaining a customer database and tracking their purchases and payments.

The shoe selection function will allow the user to search for shoes based on the type, brand, to view all shoe options or to end shopping. This feature will provide flexibility when browsing and selecting products so the user can shop based on their preferences and needs. The system will provide shoe details in a structured format making it easier to compare options and have informed decisions.

The payment function will allow users to enter a member ID and make payments towards their balance. This will simplify the payment process, resulting in efficient transactions and ensuring accurate tracking of member balances.

The discount calculation function will determine the discount percentage based on the total shoe bill. This feature encourages customers to make larger purchases by offering discounts which can boost sales and customer satisfaction. The system will automatically calculate the discount which will simplifies the checkout process, saving time and encouraging repeated purchases.

The shoe registration function will enable the addition of new shoe entries to the system including all the details such as shoe ID, type, brand, name, colour, size, quantity and price as well as updating shoe information and returning to the main menu. This feature facilitates inventory management and ensures the system stays updated with its current products.

The business information function will display details about the business such as its mission and goals, reason for establishment, and the option to display brands and return to the main menu. This feature will provide transparency and build a trusting relationship with the users by providing background information and goals of the business. The system enhances brand reputation and loyalty by showcasing its commitment to customer service.

## 

## Report Design

Users are given a bill that is outputted after they have completed shopping. The bill is printed on the screen and displays the shoes, the quantity, the cost of each individual item, discounts, if any, and finally the total price. This report is available to users who are members of the shoes store and their membership allows them to make purchases under their username and thus the bill will be charged to their individual accounts.

****

## Algorithm Design

FUNCTION main

START

Open member file (memfile) for reading

Open shoe file (shoefile) for reading

Open receipt file (recfile) for writing

mcount ← 0 // Initialise the number of members to 0 //

READ( memfile, tempID )

WHILE ( tempID <> 0 )

members[mcount].memID ← tempID

READ( members[mcount].fname, 11, memfile )

READ( members[mcount].lname, 11, memfile )

READ( members[mcount].email, 21, memfile )

READ( members[mcount].address, 25, memfile )

READ( memfile, members[mcount].gender )

READ( memfile, members[mcount].age )

READ( memfile, members[mcount].birthd )

READ( memfile, members[mcount].birthm )

READ( memfile, members[mcount].birthy )

READ( memfile, , members[mcount].balance )

mcount ← mcount + 1

READ( memfile, tempID )

ENDWHILE

scount ← 0

READ( shoefile, tempID )

WHILE ( tempID <> 0 )

shoes[scount].shoeID ← tempID

READ( shoefile, shoes[scount].brand )

FOR ( i ← 0 TO 2 )

READ( shoefile, , shoes[scount].size[i] )

FOR ( i ← 0 TO 2 )

READ( shoefile, shoes[scount].quantity[i] )

READ( shoefile, , shoes[scount].price )

READ( shoefile, shoes[scount].shoeType )

READ( shoefile, shoes[scount].gender )

READ( shoes[scount].shoeName, 25, shoefile )

READ( shoefile, shoes[scount].colour )

scount ← scount + 1

READ( shoefile, tempID )

ENDWHILE

mainmenu( members, shoes, mcount, scount )

fclose ( memfile )

fclose ( shoefile )

Open shoe file (shoefile) for reading

FOR ( tempID ← 0 TO (scount – 1) )

PRINT( shoefile, shoes[tempID].shoeID )

PRINT( shoefile, shoes[tempID].brand )

FOR ( i ← 0 to 2 )

PRINT( shoefile, shoes[tempID].size[i] )

FOR ( i ← 0 Ito 2 )

PRINT( shoefile, shoes[tempID].quantity[i] )

PRINT( shoefile, shoes[tempID].price )

PRINT( shoefile, shoes[tempID].shoeType )

PRINT( shoefile, shoes[tempID].gender )

PRINT( shoefile, shoes[tempID].shoeName )

PRINT( shoefile, shoes[tempID].colour )

ENDFOR

Open member file (memfile) for reading

FOR ( tempID ← 0 TO (mcount – 1) )

PRINT( memfile, members[tempID].memID )

PRINT( memfile, , members[tempID].fname )

PRINT( memfile, , members[tempID].lname )

PRINT( memfile, , members[tempID].email )

PRINT( memfile, , members[tempID].address )

PRINT( memfile, members[tempID].gender )

PRINT( memfile, members[tempID].age )

PRINT( memfile, members[tempID].birthd )

PRINT( memfile, members[tempID].birthm )

PRINT( memfile, members[tempID].birthy )

PRINT( memfile, members[tempID].balance )

ENDFOR

Close member file ( memfile )

Close shoe file ( shoefile )

STOP

FUNCTION printTitle

START

PRINT( " \*\*\*\*\*\* S H O E W E A R C O R N E R \*\*\*\*\*\*" )

PRINT( " \*\*\* Shoes to Match Your Mood \*\*\*" )

PRINT( " #10 X Street, Penal" )

STOP

FUNCTION printGender( int gender )

IF ( gender = 1 )

PRINT( "Female" )

ELSE

PRINT( "Male " )

STOP

FUNCTION getGender

START

// Request gender until 0 or 1 entered //

DO

printTitle

PRINT( " \*\*\*\*\*\* GENDER SELECTION \*\*\*\*\*\*" )

PRINT( " 0. Male" )

PRINT( " 1. Female" )

PRINT( " Enter the Gender # : " )

READ( gender )

WHILE (( gender < 0 ) OR ( gender > 1 ))

return gender

STOP

FUNCTION printColor( int colour )

START

IF ( colour = 1 )

PRINT( "Red " )

ELSE IF ( colour = 2 )

PRINT( "Black " )

ELSE IF ( colour = 3 )

PRINT( "Brown " )

ELSE IF ( colour = 4 )

PRINT( "Teal " )

ELSE

PRINT( "White " )

STOP

FUNCTION getColor

// Request colour until correct # entered //

DO

printTitle

PRINT( " \*\*\*\*\*\* SHOE COLOURS \*\*\*\*\*\*" )

PRINT( " 1. Red" )

PRINT( " 2. Black" )

PRINT( " 3. Brown" )

PRINT( " 4. Teal" )

PRINT( " 5. White" )

PRINT( " Enter Shoe Colour # : " )

READ( colour )

WHILE (( colour < 0 ) OR ( colour > 5 ))

return colour

STOP

FUNCTION mainmenu( memRec members[], shoeRec shoes[], int mcount, int scount )

START

DO

printTitle

PRINT( " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" )

PRINT( " \* WELCOME TO THE MAIN MENU \*" )

PRINT( " \* \*" )

PRINT( " \* 1. MEMBER OPTIONS \*" )

PRINT( " \* \*" )

PRINT( " \* 2. SHOE TRANSACTIONS \*" )

PRINT( " \* \*" )

PRINT( " \* 3. BUSINESS INFORMATION \*" )

PRINT( " \* \*" )

PRINT( " \* 4. EXIT PROGRAM \*" )

PRINT( " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" )

PRINT( " Please enter Option # : " )

READ( option )

IF ( option = 1 )

memOptions( members, shoes, mcount, scount )

ELSE IF ( option = 2 )

shoeOptions( shoes, scount )

ELSE IF ( option = 3 )

busiOption( shoes, scount )

ELSE IF ( option = 4 )

exit

ELSE

PRINT( " Invalid Option" )

Pause Output Screen

ENDIF

WHILE ( option <> 4 )

STOP

FUNCTION memOptions( memRec members[], shoeRec shoes[], int mcount, int scount )

START

DO

printTitle

PRINT( " \*\*\*\*\*\*\* MEMBER OPTIONS \*\*\*\*\*\*\*" )

PRINT( " 1. Register New Member" )

PRINT( " 2. Update/View Member Information" )

PRINT( " 3. Billing Members" )

PRINT( " 4. Make Payment" )

PRINT( " 5. Return to Main Menu" )

PRINT( " Please choose Member Option : " )

READ( option )

IF ( option = 1 )

memRegister( members, mcount)

ELSE IF ( option = 2 )

memUpdate( members, mcount )

ELSE IF ( option = 3 )

sPurchase( members, shoes, mcount, scount )

ELSE IF ( option = 4 )

payment( members, mcount )

ELSE IF ( option = 5 )

PRINT( " Returning to Main Menu" )

Pause Output Screen

ENDIF

ELSE

PRINT( " Invalid Option Number" )

Pause Output Screen

ENDIF

WHILE ( option <> 5 )

STOP

FUNCTION memRegister( memRec members[], int mcount )

START

printTitle

PRINT( " Would You like to become a Member? 1.Yes 2. No : " )

READ( regmem )

IF ( regmem = 1 )

printTitle

PRINT( " Enter First Name : " )

READ( members[mcount].fname, 11, stdin )

PRINT( " Enter Last Name : " )

READ( members[mcount].lname, 11, stdin )

PRINT( " Email Address : " )

READ( members[mcount].email, 21, stdin )

PRINT( " Enter Address : " )

READ( members[mcount].address, 25, stdin )

PRINT( " Enter Member Age : " )

READ( members[mcount].age )

PRINT( " Enter Birth Day : " )

READ( members[mcount].birthd )

PRINT( " Enter Birth Month : " )

READ( members[mcount].birthm )

PRINT( " Enter Birth Year : " )

READ( members[mcount].birthy )

members[mcount].gender ← getGender

members[mcount].memID ← mcount + 1

members[mcount].balance ← 0

printTitle

PRINT( " First Name : ", members[mcount].fname )

PRINT( " Last Name : ", members[mcount].lname )

PRINT( " Email Address : ", members[mcount].email )

PRINT( " Address : ", members[mcount].address )

PRINT( " Member Age : ", members[mcount].age )

PRINT( " Date of Birth : ", members[mcount].birthd, members[mcount].birthm, members[mcount].birthy )

PRINT( " Gender : " )

printGender( members[mcount].gender )

members[mcount].memID ← mcount + 1

members[mcount].balance ← 0

PRINT( " Your have been registered. You are a Honarary Member!!!" )

PRINT( " Your Member ID is ", members[mcount].memID )

mcount ← mcount + 1

ELSE IF ( regmem = 2 )

PRINT( " Return to Main Menu" )

ELSE

PRINT( " Invalid Action" )

ENDIF

Pause Output Screen

STOP

FUNCTION memUpdate( memRec members[], int mcount )

START

printTitle

PRINT( " Enter Member ID : " )

READ( tmemid )

IF (( tmemid > 0 ) AND ( tmemid <= mcount ))

PRINT( " MEMBER NAME - ", members[tmemid - 1].fname, members[tmemid - 1].lname )

PRINT( " Would you like to Update the Information? " )

PRINT( "1. Yes 2. No : " )

READ( y\_n )

IF ( y\_n = 1 )

printTitle

PRINT( " UPDATE MEMBER INFORMATION" )

PRINT( " ============================" )

PRINT( " MEMBER NAME - ", members[tmemid - 1].fname, members[tmemid - 1].lname )

PRINT( " EMAIL ADDRESS - ", members[tmemid - 1].email )

PRINT( " ADDRESS - ", members[tmemid - 1].address )

PRINT( " BALANCE - $", members[tmemid - 1].balance )

PRINT( " 1. Last Name" )

PRINT( " 2. Email Address" )

PRINT( " 3. Bill Address" )

PRINT( " 4. No Changes" )

PRINT( " Enter Update # : " )

READ( memup )

IF ( memup = 1 )

PRINT( " Enter New Last Name : " )

READ( members[tmemid - 1].lname )

ELSE IF ( memup = 2 )

PRINT( " Enter New Email Address : " )

READ( members[tmemid - 1].email )

ELSE IF ( memup = 3 )

PRINT( " Enter New Bill Address : " )

READ( members[tmemid - 1].address - 1, 25, stdin )

ELSE IF ( memup = 4 )

PRINT( " Customer Information Viewed" )

ELSE

PRINT( " Invalid Customer Update Choice" )

ENDIF

IF (( memup >= 1 ) AND ( memup < 4 ))

PRINT( " The information has been updated!!!!" )

ELSE IF ( y\_n = 2 )

PRINT( " Return to Main Menu" )

ELSE

PRINT( " Invalid Action" )

ENDIF

ELSE

PRINT( " Invalid Customer ID" )

Pause Output Screen

STOP

FUNCTION sPurchase( memRec members[], shoeRec shoes[], int mcount, int scount )

START

Open dill file (billfile) for writing

printTitle

PRINT( " Enter Member ID (0 for Non-Member) : " )

READ( tmemid )

IF (( tmemid >= 0 ) AND ( tmemid <= mcount ))

IF ( tmemid <> 0 )

printTitle

PRINT( " Member Id : ", tmemid )

PRINT( " Member Name : ", members[tmemid - 1].fname, members[tmemid - 1].lname )

Pause Output Screen

ENDIF

shoeBill ← 0

numshoes ← 0

PRINT( billfile, " \*\*\*\*\*\* S H O E W E A R C O R N E R \*\*\*\*\*\*" )

PRINT( billfile, " #10 X Street, Penal" )

PRINT( billfile, " SHOE QUANTITY PRICE TOTAL PRICE" )

// Get all the shoes purchased to produce total shoe bill //

DO

// Get the shoe id to purchase //

tshoeid ← getShoe( shoes, scount )

IF ( tshoeid <> 0 )

DO

// Print the three sizes available //

printTitle

PRINT( " -----------------" )

PRINT( " | NO. | SHOE SIZE |" )

FOR ( i ← 0 TO 2 )

PRINT( " |-----|-----------|" )

PRINT( " | “, i, “ | “, shoes[tshoeid - 1].size[i] )

ENDFOR

PRINT( " |-----|-----------|" )

PRINT( " | “, I, | NONE |" )

PRINT( " ------------------" )

PRINT( " Enter Shoe Size : " )

READ( ssize )

WHILE (( ssize < 0 ) OR ( ssize > 3 ))

PRINT( " Enter Quantity required (0 IF none) : " )

READ( squan )

// There is a size that the user wants //

IF (( squan <> 0 ) AND ( ssize <> 0 ))

IF ( squan <= shoes[tshoeid - 1].quantity[ssize] )

// Increase the number of shoes in this purchase //

numshoes ← numshoes + squan

shoeCost ← squan \* shoes[tshoeid - 1].price

shoeBill ← shoeBill + shoeCost

shoes[tshoeid - 1].quantity[ssize] ← shoes[tshoeid - 1].quantity[ssize] - squan

PRINT( billfile, shoes[tshoeid - 1].shoeName, squan, shoes[tshoeid - 1].price, shoeCost )

ENDIF

ELSE

PRINT( " Insufficient stock to fill order" )

Pause Output Screen

ENDIF

ELSE

PRINT( " No Size for the Customer" )

ENDIF

WHILE ( tshoeid <> 0 )

IF ( shoeBill <> 0.00 )

printTitle

PRINT( " SHOE BILL - $", shoeBill )

IF ( tmemid <>0 )

discount ← calcuDis( shoeBill )

disAmount ← shoeBill \* discount/100

finalBill ← shoeBill - disAmount

PRINT( " TOTAL BILL - $", shoeBill )

PRINT( " DISCOUNT - ", discount )

PRINT( "$", disAmount )

PRINT( billfile, " TOTAL BILL - $", shoeBill )

PRINT( billfile, " DISCOUNT - ", discount )

PRINT( billfile, "$", disAmount )

ELSE

finalBill ← shoebill

ENDIF

PRINT( " FINAL BILL - $", finalBill )

PRINT( billfile, " FINAL BILL - $", finalBill )

Pause Output Screen

members[tmemid - 1].balance ← members[tmemid - 1].balance + finalBill

ENDIF

fclose( billfile )

STOP

FUNCTION payment( memRec members[], int mcount )

START

printTitle

PRINT( " Enter the Member ID : " )

READ( tmemid )

IF (( tmemid > 0 ) AND ( tmemid < mcount ))

memloc ← tmemid - 1

printTitle

PRINT( " MEMBER INFORMATION" )

PRINT( " ====================" )

PRINT( " MEMBER ID : ", tmemid )

PRINT( " MEMBER NAME : ", members[tmemid - 1].fname, members[tmemid - 1].lname )

PRINT( " BALANCE : $", members[tmemid - 1].balance )

IF ( members[memloc].balance = 0 )

PRINT( " No outstanding balance!!!" )

ELSE

PRINT( " Enter Payment : $" )

READ( pd )

IF ( pd < members[memloc].balance )

members[memloc].balance ← members[memloc].balance - pd

ELSE IF ( pd > members[tmemid - 1].balance )

change ← pd - members[memloc].balance

members[memloc].balance ← 0.00

PRINT( " Change : $", change )

ELSE

members[tmemid - 1].balance ← 0.00

PRINT( " NEW BALANCE : $", members[memloc].balance )

ENDIF

ENDIF

Pause Output Screen

STOP

FUNCTION calcuDis( float shoeBill )

START

IF ( shoeBill > 2000.00 )

return 20

ELSE

return 0

STOP

FUNCTION getShoe( shoeRec shoes[], int shoecount )

START

Open receipt file (recfile) for writing

PRINT( recfile, " \*\*\*\*\*\* S H O E W E A R C O R N E R \*\*\*\*\*\*" )

PRINT( recfile, " \*\*\* Shoes to Match Your Mood \*\*\*" )

PRINT( recfile, " #10 X Street, Penal" )

printTitle

PRINT( " SEARCH CRITERIA" )

PRINT( " ================" )

PRINT( " 1. SHOE TYPE" )

PRINT( " 2. SHOE BRAND" )

PRINT( " 3. ALL SHOES" )

PRINT( " 4. END SHOPPING" )

PRINT( " Enter the Search # : " )

READ( search )

IF ( search = 1 )

shoet ← getShoeType

ELSE IF ( search = 2 )

shoeb ← getShoeBrand

ELSE IF ( search = 3 )

shoet ← 0

shoeb ← 0

ELSE IF ( search = 4 )

PRINT( " Your bill would be printed" )

Pause Output Screen

tshoeid ← 0

ELSE

PRINT( " Invalid Search Choice" )

Pause Output Screen

ENDIF

printTitle

IF (( search >= 1 ) AND ( search < 4 ))

// Indicates IF any shoes purchased to not print empty list at the end //

numshoep ← 0

IF ( search = 1 )

printTitle

PRINT( " SHOE TYPE (" )

printShoeType( shoet )

PRINT( " SEARCH" )

PRINT( " ---------------------------------------------------------------- " )

PRINT( "| SHOE | SHOE NAME | SHOE | GENDER | PRICE |" )

PRINT( "| ID | | BRAND | | |" )

PRINT( "|------|--------------------------|----------|--------|----------|" )

ELSE IF ( search = 2 )

printTitle

PRINT( " SHOE BRAND (" )

printBrand( shoeb )

PRINT( " SEARCH" )

PRINT( " ---------------------------------------------------------------- " )

PRINT( "| SHOE | SHOE NAME | SHOE | GENDER | PRICE |" )

PRINT( "| ID | | TYPE | | |" )

PRINT( "|------|--------------------------|----------|--------|----------|" )

ELSE

printTitle

PRINT( " ALL SHOES SEARCH" )

PRINT( "|\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|" )

PRINT( " --------------------------------------------------------------------------- " )

PRINT( "| SHOE | SHOE NAME | SHOE | BRAND | GENDER | PRICE |" )

PRINT( "| ID | | TYPE | | | |" )

PRINT( "|------|--------------------------|----------|----------|--------|-----------|" )

ENDIF

FOR ( i ← 0 TO (shoecount – 1) )

IF ( search = 1 )

IF ( shoet = shoes[i].shoeType )

PRINT( shoes[i].shoeID, shoes[i].shoeName )

PRINT( "| " )

printBrand( shoes[i].brand )

numshoep ← numshoep + 1

IF ( shoes[i].gender = 1 )

PRINT( " | Female " )

ELSE

PRINT( " | Male " )

PRINT( "| “, shoes[i].price, “ |",)

PRINT( "| “, shoes[i].shoeID, “ |”, shoes[i].shoeName )

ENDIF

ELSE IF ( search = 2 )

IF ( shoeb = shoes[i].brand )

PRINT( "| “, shoes[i].shoeID, “ |”, shoes[i].shoeName )

PRINT( "| " )

printShoeType( shoes[i].shoeType )

numshoep ← numshoep + 1

IF ( shoes[i].gender = 1 )

PRINT( " | Female " )

ELSE

PRINT( " | Male " )

PRINT( "| $”, shoes[i].price. “ |")

ENDIF

ELSE

PRINT( "| “, shoes[i].shoeID, “ | “, [i].shoeName )

PRINT( "| " )

printBrand( shoes[i].brand )

PRINT( " | " )

printShoeType( shoes[i].shoeType )

// Checks the number of shoes printed FOR search //

numshoep ← numshoep + 1

IF ( shoes[i].gender = 1 )

PRINT( " | Female " )

ELSE

PRINT( " | Male " )

PRINT( "| $”, shoes[i].price, “ |")

ENDIF

ENDFOR

IF (( search = 1 ) OR ( search = 2 ))

PRINT( "|\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|" )

ELSE

PRINT( "|\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|" )

IF ( numshoep = 0 )

printTitle

PRINT( " NO SHOES AVAILABLE" )

Pause Output Screen

ELSE

PRINT( " Enter the Shoe ID to Purchase (0 to stop) : " )

READ( tshoeid )

ENDIF

IF ( search = 1 )

IF ( shoes[tshoeid - 1].shoeType = shoet )

return tshoeid

ELSE

PRINT( " You must enter Shoe No for type selected" )

Pause Output Screen

return 0

ENDIF

ENDIF

ELSE IF ( search = 2 )

IF ( shoes[tshoeid - 1].brand = shoeb )

return tshoeid

ELSE

PRINT( " You must enter Shoe No FOR Brand selected" )

Pause Output Screen

return 0

ENDIF

ELSE IF ( search = 3 )

return tshoeid

ELSE

return 0

STOP

FUNCTION shoeOptions( shoeRec shoes[], int shoecount )

START

do

printTitle

PRINT( " \*\*\*\*\*\*\* SHOE OPTIONS \*\*\*\*\*\*\*" )

PRINT( " 1. Register New Shoe" )

PRINT( " 2. Update Shoe InFORmation" )

PRINT( " 3. Return to Main Menu" )

PRINT( " Enter Shoe Option # : " )

READ( shoeopt )

IF ( shoeopt = 1 )

shoeRegister( shoes, shoecount )

ELSE IF ( shoeopt = 2 )

shoeUpdate( shoes, shoecount )

ELSE

PRINT( "Return to Main Menu" )

Pause Output Screen

ENDIF

WHILE ( shoeopt <> 3 )

STOP

FUNCTION shoeRegister( shoeRec shoes[], int shoecount )

START

printTitle

PRINT( " \*\*\*\*\*\* SHOE REGISTRATION \*\*\*\*\*\*" )

shoes[shoecount].shoeType ← getShoeType

shoes[shoecount].brand ← getShoeBrand

PRINT( " Enter Shoe Name : " )

READ( shoes[shoecount].shoeName, 25, stdin )

READ( shoes[shoecount].shoeName, 25, stdin )

shoes[shoecount].colour ← getColor

printTitle

FOR ( i ← 0 TO 2 )

PRINT( " Enter Shoe Size : " )

READ( shoes[shoecount].size[i] )

PRINT( " Enter Quantity : " )

READ( shoes[shoecount].quantity[i] )

ENDFOR

PRINT( " Enter Shoe Price : $" )

READ( , shoes[shoecount].price )

printTitle

PRINT( " NEW SHOE INFORMATION" )

PRINT( " ======================" )

PRINT( " SHOE ID - ", shoecount )

PRINT( " SHOE TYPE - " )

printShoeType( shoes[shoecount].shoeType )

PRINT( " SHOE NAME - ", shoes[shoecount].shoeName )

PRINT( " SHOE BRAND - " )

printBrand( shoes[shoecount].brand )

PRINT( " COLOUR - " )

printColor( shoes[shoecount].colour )

PRINT( " SHOE SIZE QUANTITY" )

FOR ( i ← 0 TO 2 )

PRINT( shoes[shoecount].size[i], shoes[shoecount].quantity[i] )

PRINT( " PRICE - $", shoes[shoecount].price )

PRINT( " The new shoe has been registered." )

shoecount ← shoecount + 1

shoes[shoecount].shoeID ← shoecount

Pause Output Screen

STOP

FUNCTION shoeUpdate( shoeRec shoes[], int shoecount )

START

printTitle

PRINT( " Enter Shoe ID : " )

READ( tshoeid )

IF ( ( tshoeid > 0 ) AND ( tshoeid < shoecount ))

printTitle

PRINT( " SHOE INFORMATION" )

PRINT( " ==================" )

PRINT( " SHOE ID - ", tshoeid )

PRINT( " SHOE NAME - ", shoes[tshoeid - 1].shoeName )

PRINT( " SHOE PRICE - $", shoes[tshoeid - 1].price )

PRINT( " SHOE SIZE STOCK AMOUNT" )

FOR ( i ← 0 TO 2 )

PRINT( shoes[tshoeid - 1].size[i], shoes[tshoeid - 1].quantity[i] )

PRINT( " 1. Quantity Size", shoes[tshoeid - 1].size[0], shoes[tshoeid - 1].quantity[0] )

PRINT( " 2. Quantity Size", shoes[tshoeid - 1].size[1], shoes[tshoeid - 1].quantity[1] )

PRINT( " 3. Quantity Size", shoes[tshoeid - 1].size[2], shoes[tshoeid - 1].quantity[2] )

PRINT( " 4. Shoe Price" )

PRINT( " 5. No Change" )

PRINT( " Enter the information # to change : " )

READ( change )

IF (( change = 1 ) OR ( change = 2 ) OR ( change = 3 ))

PRINT( " Enter the New Quantity : " )

READ( quan )

shoes[tshoeid - 1].quantity[change - 1] ← shoes[tshoeid - 1].quantity[change - 1] + quan

printTitle

PRINT( " SHOE ID - ", tshoeid )

PRINT( " SHOE NAME - ", shoes[tshoeid - 1].shoeName )

PRINT( " SHOE PRICE - $", shoes[tshoeid - 1].price )

PRINT( " SHOE SIZE STOCK AMOUNT" )

FOR ( i ← 0 TO 4 )

PRINT( shoes[tshoeid - 1].size[i], shoes[tshoeid - 1].quantity[i] )

PRINT( " The shoe quantity has been successfully updated." )

ELSE IF ( change = 4 )

PRINT( " Enter the New Shoe Price : $" )

READ( shoes[tshoeid - 1].price )

PRINT( " The shoe price has been successfully updated." )

ELSE

PRINT( " Thanks for viewing information " )

ENDIF

ELSE

PRINT( " Invalid choice, only change price and quantity" )

Pause Output Screen

STOP

FUNCTION busiOption( shoeRec shoes[], int shoecount )

START

DO

printTitle

PRINT( " \*\*\*\*\*\* BUSINESS OPTIONS \*\*\*\*\*\*" )

PRINT( "1. About Business" )

PRINT( "2. Display Brands" )

PRINT( "3. Return to Main Menu" )

PRINT( "Please enter option : " )

READ( choice )

IF ( choice = 1 )

busiAbt

ELSE IF ( choice = 2 )

brand ← getShoeBrand

ELSE IF ( choice = 3 )

PRINT( "Returning to Main Menu" )

Pause Output Screen

ELSE

PRINT( "Invalid Option" )

Pause Output Screen

ENDIF

WHILE ( choice <> 3 )

STOP

FUNCTION busiAbt

START

Open company information file (abtfile) for reading

IF ( !abtfile )

PRINT( "Cannot open file" )

ELSE

printTitle

linecount ← 1

WHILE ( READ( infoline, sizeof(infoline), abtfile ))

PRINT( , infoline )

IF (( linecount MOD 5 = 0 ))

PRINT("")

Pause Output Screen

printTitle

ENDIF

linecount++

ENDWHILE

Pause Output Screen

STOP

FUNCTION getShoeType

START

printTitle

PRINT( " SHOE CATEGORIES" )

PRINT( " ================" )

PRINT( " 1. Athletic Shoes and Sneakers" )

PRINT( " 2. Heels Pumps" )

PRINT( " 3. Wedges" )

PRINT( " 4. Designer Shoes" )

PRINT( " 5. Sandals" )

PRINT( " 6. Slippers" )

PRINT( " 7. Slides Mules" )

PRINT( " 8. Dress Shoes" )

PRINT( " 9. Boots" )

PRINT( " 10. NEW ARRIVALS" )

PRINT( " Please enter Shoe Category # : " )

READ( shoetype )

return shoetype

STOP

FUNCTION printShoeType( int shoetype )

START

IF ( shoetype = 1 )

PRINT( "Sneakers" )

ELSE IF ( shoetype = 2 )

PRINT( "Heels " )

ELSE IF ( shoetype = 3 )

PRINT( "Wedges " )

ELSE IF ( shoetype = 4 )

PRINT( "Designer" )

ELSE IF ( shoetype = 5 )

PRINT( "Sandals " )

ELSE IF ( shoetype = 6 )

PRINT( "Slippers" )

ELSE IF ( shoetype = 7 )

PRINT( "Slides " )

ELSE IF ( shoetype = 8 )

PRINT( "Dress " )

ELSE IF ( shoetype = 9 )

PRINT( "Boots " )

ELSE

PRINT( "New " )

STOP

FUNCTION getShoeBrand

START

DO

printTitle

PRINT( " BRANDS AVAILABLE" )

PRINT( " ==================" )

PRINT( " -------------------------- " )

PRINT( " | BRAND | BRAND NAME |" )

PRINT( " |-------|------------------|" )

PRINT( " | 1. | Adidas |" )

PRINT( " | 2. | Nike |" )

PRINT( " | 3. | Reebok |" )

PRINT( " | 4. | Puma |" )

PRINT( " | 5. | Gucci |" )

PRINT( " | 6. | Fila |" )

PRINT( " | 7. | Louis Vuitton |" )

PRINT( " | 8. | Champion |" )

PRINT( " | 9. | Clarks |" )

PRINT( " | 10. | Fioni |" )

PRINT( " -------------------------- " )

PRINT( " 0. NO BRAND OR VIEWED BRANDS" )

PRINT( " Enter Brand # : " )

READ( brand )

WHILE (( brand < 0 ) OR ( brand > 10 ))

return brand

STOP

FUNCTION printBrand( int brand )

START

IF ( brand = 1 )

PRINT( "Adidas " )

ELSE IF ( brand = 2 )

PRINT( "Nike " )

ELSE IF ( brand = 3 )

PRINT( "Reebok " )

ELSE IF ( brand = 4 )

PRINT( "Puma " )

ELSE IF ( brand = 5 )

PRINT( "Gucci " )

ELSE IF ( brand = 6 )

PRINT( "Fila " )

ELSE IF ( brand = 7 )

PRINT( "Madden " )

ELSE IF ( brand = 8 )

PRINT( "Champion" )

ELSE IF ( brand = 9 )

PRINT( "Clarks " )

ELSE

PRINT( "Fioni " )

ENDIF

STOP

## Data Structures

Data structure refers to a data organisation, management and storage format that enables the system to run efficiently and allows for easy changes and modifications not be made. For the system, record structures which are collection of related items to be treated as a unit in the program. This was utilized for the creation of custom storage locations with different data types for ease of manipulation of data within the system. In addition, several internet and float data structures were used throughout the program. These structures, their data types and a brief description of each is given in the following tables.

**memRec**

|  |  |  |
| --- | --- | --- |
| FIELD NAME | DATA TYPE | DESCRIPTION |
| memID | Integer | Stores member’s identification number |
| fname | Array of eleven characters | Stores member’s first name |
| lname | Array of eleven characters | Stores member’s last name |
| email | Array of twenty one characters | Stores member’s email address |
| address | Array of twenty five characters | Stores member’s house number, street number and city/town |
| gender | Integer | Stores member’s gender (1-female 2-male) |
| age | Integer | Stores member’s age in years |
| birthd | Integer | Stores member’s day of birth |
| birthm | Integer | Stores member’s month of birth |
| birthy | Integer | Stores member’s year of birth |
| balance | Float | Stores member’s balance |

**shoeRec**

|  |  |  |
| --- | --- | --- |
| FIELD NAME | DATA TYPE | DESCRIPTION |
| shoeID | Integer | Stores shoe’s identification number |
| brand | Integer | Stores shoe’s brand |
| size | Array of three Float | Stores shoe’s sizes |
| quantity | Array of three Integer | Stores amount of shoes for each size |
| price | Float | Stores shoe’s price |
| shoeType | Integer | Stores shoe’s type |
| gender | Integer | Stores gender for shoe |
| shoeName | Array of twenty-five characters | Stores shoe title |
| colour | Integer | Stores colour of shoe |

# Variable Table

|  |  |  |
| --- | --- | --- |
| VARIBLE NAME | DATA TYPE | DESCRIPTION |
| members[50] | Array of fifty memRec | Stores records for fifty members |
| shoes[100] | Array of one hundred shoeRec | Stores records for one hundred shoes |
| tempID | Integer | Stores temporary member ID/shoe ID |
| mcount | Integer | Stores amount of members |
| scount | Integer | Stores the amount of shoes |
| gender | Integer | Stores the gender (0-male 1-female) |
| colour | Integer | Stores the colour of the clothing |
| option | Integer | Stores customer’s choice in Main Menu |
| category | Integer | Stores member’s choice in member options |
| regmem | Integer | Stores confirmation that the new member is registering |
| y\_n | Integer | Stores update verification number |
| tmemID | Integer | Stores temporary member ID |
| discount | Float | Stores member’s discounted percentage |
| squan | Integer | Stores quantity of a particular shoe purchased |
| shoeBill | Float | Stores member’s shoe bill |
| numShoes | Integer | Stores number of shoes purchased |
| disAmount | Float | Stores the amount of discount |
| finalBill | Float | Stores final shoe bill |
| pd | Float | Stores the amount paid by member |
| change | Float | Stores difference between the amount |
| memloc | Integer | Stores location of member record in the file |
| tshoeid | Integer | Stores temporary shoe ID |
| shoet | Integer | Stores shoe type |
| numshoep | Integer | Stores total amount of shoes purchased |
| shoeopt | Integer | Stores shoe option chosen |
| choicechnge | Integer | Stores the number of shoe field to be updated |
| quan | Integer | Stores the number of shoes |
| choice | Integer | Stores the business option choice |
| shoetype | Integer | Stores the type of shoe chosen by user |
| brand | Integer | Stores the number of the brand |

# Coding and Testing

## Code

/\* STUDENT NAME : Eliana Dookhoo 1600260290

Khadine Rambert 1600260419

Mikayla S. Roberts 1600260443

Sharona Lochan 1600260141

CENTER : HOLY FAITH CONVENT PENAL

PROJECT TITLE : SOFTWARE FOR SHOEWEAR CORNER

PURPOSE : Perform functions read and write to file, search, add/update/remove records

\*/

#include <stdio.h>

#include <string.h>

#include <dos.h>

#define VAT 12.5

typedef struct

{

int memID;

char fname[11];

char lname[11];

char email[21];

char address[25];

int gender;

int age;

int birthd;

int birthm;

int birthy;

float balance;

} memRec;

typedef struct

{

int shoeID;

int brand;

float size[3];

int quantity[3];

float price;

int shoeType;

int gender;

char shoeName[25];

int colour;

} shoeRec;

void main()

{

void printTitle();

void printColors( int );

int getColor();

void printGender( int );

int getGender();

void mainmenu( memRec[], shoeRec[], int, int );

void memOptions( memRec[], shoeRec[], int, int );

void memRegister( memRec[], int );

void memUpdate( memRec[], int );

void sPurchase( memRec[], shoeRec[], int, int );

void payment( memRec[], int );

int calcuDis( float );

void shoeOptions( shoeRec[], int );

void shoeRegister( shoeRec[], int );

void shoeUpdate( shoeRec[], int );

int getShoeType();

void printShoeType( int );

void busiOption( shoeRec[], int );

void busiAbt();

void busiBd( shoeRec[], int );

int getShoeBrand();

void printBrand( int );

memRec members[50];

shoeRec shoes[100];

int i;

int tempID;

int mcount;

int scount;

FILE \*memfile;

FILE \*shoefile;

FILE \*recfile;

memfile = fopen( "MEMK.TXT", "r" );

shoefile = fopen( "Shoel.txt", "r" );

mcount = 0; /\* Initialise the number of members to 0 \*/

fscanf( memfile, "%d", &tempID );

while ( tempID != 0 )

{

members[mcount].memID = tempID;

fgets( members[mcount].fname, 11, memfile );

fgets( members[mcount].lname, 11, memfile );

fgets( members[mcount].email, 21, memfile );

fgets( members[mcount].address, 25, memfile );

fscanf( memfile, "%d", &members[mcount].gender );

fscanf( memfile, "%d", &members[mcount].age );

fscanf( memfile, "%d", &members[mcount].birthd );

fscanf( memfile, "%d", &members[mcount].birthm );

fscanf( memfile, "%d", &members[mcount].birthy );

fscanf( memfile, "%f", &members[mcount].balance );

mcount = mcount + 1;

fscanf( memfile, "%d", &tempID );

}

scount = 0;

fscanf( shoefile, "%d", &tempID );

while ( tempID != 0 )

{

shoes[scount].shoeID = tempID;

fscanf( shoefile, "%d", &shoes[scount].brand );

for ( i = 0; i < 3; i++ )

fscanf( shoefile, "%f", &shoes[scount].size[i] );

for ( i = 0; i < 3; i++ )

fscanf( shoefile, "%d", &shoes[scount].quantity[i] );

fscanf( shoefile, "%f", &shoes[scount].price );

fscanf( shoefile, "%d", &shoes[scount].shoeType );

fscanf( shoefile, "%d", &shoes[scount].gender );

fgets( shoes[scount].shoeName, 25, shoefile );

fscanf( shoefile, "%d", &shoes[scount].colour );

scount = scount + 1;

fscanf( shoefile, "%d", &tempID );

}

mainmenu( members, shoes, mcount, scount );

fclose ( memfile );

fclose ( shoefile );

shoefile = fopen( "Shoel.txt", "w" );

for ( tempID = 0; tempID < scount; tempID++ )

{

fprintf( shoefile, "%d ", shoes[tempID].shoeID );

fprintf( shoefile, "%d ", shoes[tempID].brand );

for ( i = 0; i < 3; i++ )

fprintf( shoefile, "%6.2f ", shoes[tempID].size[i] );

for ( i = 0; i < 3; i++ )

fprintf( shoefile, "%d ", shoes[tempID].quantity[i] );

fprintf( shoefile, "%6.2f ", &shoes[tempID].price );

fprintf( shoefile, "%d ", shoes[tempID].shoeType );

fprintf( shoefile, "%d", shoes[tempID].gender );

fprintf( shoefile, "%s ", shoes[tempID].shoeName );

fprintf( shoefile, "%d\n", shoes[tempID].colour );

}

memfile = fopen( "MEMK.TXT", "w" );

for ( tempID = 0; tempID < mcount; tempID++ )

{

fprintf( memfile, "%d", members[tempID].memID );

fprintf( memfile, "%s", members[tempID].fname );

fprintf( memfile, "%s", members[tempID].lname );

fprintf( memfile, "%s", members[tempID].email );

fprintf( memfile, "%s", members[tempID].address );

fprintf( memfile, "%d ", members[tempID].gender );

fprintf( memfile, "%d ", members[tempID].age );

fprintf( memfile, "%d ", members[tempID].birthd );

fprintf( memfile, "%d ", members[tempID].birthm );

fprintf( memfile, "%d ", members[tempID].birthy );

fprintf( memfile, "%6.2f\n", members[tempID].balance );

}

fclose( memfile );

fclose( shoefile );

}

void printTitle()

{

system( "cls" );

printf( " \*\*\*\*\*\* S H O E W E A R C O R N E R \*\*\*\*\*\*\n" );

printf( " \*\*\* Shoes to Match Your Mood \*\*\*\n" );

printf( " #10 X Street, Penal\n\n" );

}

void printGender( int gender )

{

if ( gender == 1 )

printf( "Female" );

else

printf( "Male " );

}

int getGender()

{

int gender;

/\* Request gender until 0 or 1 entered \*/

do

{

printTitle();

printf( " \*\*\*\*\*\* GENDER SELECTION \*\*\*\*\*\*\n\n" );

printf( " 0. Male\n\n" );

printf( " 1. Female\n\n\n" );

printf( " Enter the Gender # : " );

scanf( "%d", &gender );

}

while (( gender < 0 ) || ( gender > 1 ));

return gender;

}

void printColor( int colour )

{

if ( colour == 1 )

printf( "Red " );

else if ( colour == 2 )

printf( "Black " );

else if ( colour == 3 )

printf( "Brown " );

else if ( colour == 4 )

printf( "Teal " );

else

printf( "White " );

}

int getColor()

{

int colour;

/\* Request colour until correct # entered \*/

do

{

printTitle();

printf( " \*\*\*\*\*\* SHOE COLOURS \*\*\*\*\*\*\n\n" );

printf( " 1. Red\n" );

printf( " 2. Black\n" );

printf( " 3. Brown\n" );

printf( " 4. Teal\n" );

printf( " 5. White\n\n" );

printf( " Enter Shoe Colour # : " );

scanf( "%d", &colour );

}

while (( colour < 0 ) || ( colour > 5 ));

return colour;

}

void mainmenu( memRec members[], shoeRec shoes[], int mcount, int scount )

{

int option;

do

{

printTitle();

printf( " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n" );

printf( " \* WELCOME TO THE MAIN MENU \*\n" );

printf( " \* \*\n" );

printf( " \* 1. MEMBER OPTIONS \*\n" );

printf( " \* \*\n" );

printf( " \* 2. SHOE TRANSACTIONS \*\n" );

printf( " \* \*\n" );

printf( " \* 3. BUSINESS INFORMATION \*\n" );

printf( " \* \*\n" );

printf( " \* 4. EXIT PROGRAM \*\n" );

printf( " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n" );

printf( " Please enter Option # : " );

scanf( "%d", &option );

if ( option == 1 )

memOptions( members, shoes, mcount, scount );

else if ( option == 2 )

shoeOptions( shoes, scount );

else if ( option == 3 )

busiOption( shoes, scount );

else if ( option == 4 )

exit();

else

{

printf( "\n Invalid Option\n\n" );

system( "pause" );

}

}

while ( option != 4 );

}

void memOptions( memRec members[], shoeRec shoes[], int mcount, int scount )

{

int option;

do

{

printTitle();

printf( " \*\*\*\*\*\*\* MEMBER OPTIONS \*\*\*\*\*\*\*\n\n" );

printf( " 1. Register New Member\n\n" );

printf( " 2. Update/View Member Information\n\n" );

printf( " 3. Billing Members\n\n" );

printf( " 4. Make Payment\n\n" );

printf( " 5. Return to Main Menu\n\n\n" );

printf( " Please choose Member Option : " );

scanf( "%d", &option );

if ( option == 1 )

memRegister( members, mcount);

else if ( option == 2 )

memUpdate( members, mcount );

else if ( option == 3 )

sPurchase( members, shoes, mcount, scount );

else if ( option == 4 )

payment( members, mcount );

else if ( option == 5 )

{

printf( "\n Returning to Main Menu\n\n" );

system( "pause" );

}

else

{

printf( "\n Invalid Option Number\n\n" );

system( "pause" );

}

}

while ( option != 5 );

}

void memRegister( memRec members[], int mcount )

{

int regmem;

int length;

int i;

printTitle();

printf( " Would You like to become a Member? 1.Yes 2. No : " );

scanf( "%d", &regmem );

if ( regmem == 1 )

{

printTitle();

printf( " Enter First Name : " );

fgets( members[mcount].fname, 11, stdin );

printf( "\n Enter Last Name : " );

fgets( members[mcount].lname, 11, stdin );

printf( "\n Email Address : " );

fgets( members[mcount].email, 21, stdin );

printf( "\n Enter Address : " );

fgets( members[mcount].address, 25, stdin );

printf( "\n Enter Member Age : " );

scanf( "%d", &members[mcount].age );

printf( "\n Enter Birth Day : " );

scanf( "%d", &members[mcount].birthd );

printf( "\n Enter Birth Month : " );

scanf( "%d", &members[mcount].birthm );

printf( "\n Enter Birth Year : " );

scanf( "%d", &members[mcount].birthy );

members[mcount].gender = getGender();

members[mcount].memID = mcount + 1;

members[mcount].balance = 0;

printTitle();

printf( " First Name : %s\n", members[mcount].fname );

printf( " Last Name : %s\n", members[mcount].lname );

printf( " Email Address : %s\n", members[mcount].email );

printf( " Address : %s\n", members[mcount].address );

printf( " Member Age : %d\n", members[mcount].age );

printf( " Date of Birth : %d/%d/%d\n", members[mcount].birthd, members[mcount].birthm, members[mcount].birthy );

printf( " Gender : " );

printGender( members[mcount].gender );

members[mcount].memID = mcount + 1;

members[mcount].balance = 0;

printf( "\n\n Your have been registered. You are a Honarary Member!!!" );

printf( "\n\n Your Member ID is %d\n\n", members[mcount].memID );

mcount = mcount + 1;

}

else if ( regmem == 2 )

printf( "\n Return to Main Menu\n\n" );

else

printf( "\n Invalid Action\n\n" );

system( "pause" );

}

void memUpdate( memRec members[], int mcount )

{

int y\_n;

int memup;

int tmemid;

printTitle();

printf( " Enter Member ID : " );

scanf( "%d", &tmemid );

if (( tmemid > 0 ) && ( tmemid <= mcount ))

{

printf( "\n MEMBER NAME - %s %s\n\n", members[tmemid - 1].fname, members[tmemid - 1].lname );

printf( " Would you like to Update the Information? " );

printf( "1. Yes 2. No : " );

scanf( "%d", &y\_n );

if ( y\_n == 1 )

{

printTitle();

printf( " UPDATE MEMBER INFORMATION\n" );

printf( " ==========================\n\n" );

printf( " MEMBER NAME - %s %s\n\n", members[tmemid - 1].fname, members[tmemid - 1].lname );

printf( " EMAIL ADDRESS - %s\n\n", members[tmemid - 1].email );

printf( " ADDRESS - %s\n\n", members[tmemid - 1].address );

printf( " BALANCE - $%7.2f\n\n", members[tmemid - 1].balance );

printf( " 1. Last Name\n" );

printf( " 2. Email Address\n" );

printf( " 3. Bill Address\n" );

printf( " 4. No Changes\n\n" );

printf( " Enter Update # : " );

scanf( "%d", &memup );

if ( memup == 1 )

{

printf( "\n Enter New Last Name : " );

scanf( "%s", members[tmemid - 1].lname );

}

else if ( memup == 2 )

{

printf( "\n Enter New Email Address : " );

scanf( "%s", members[tmemid - 1].email );

}

else if ( memup == 3 )

{

printf( "\n Enter New Bill Address : " );

fgets( members[tmemid - 1].address - 1, 25, stdin );

}

else if ( memup == 4 )

{

printf( "\n Customer Information Viewed\n\n" );

}

else

printf( "\n Invalid Customer Update Choice\n\n" );

if (( memup >= 1 ) && ( memup < 4 ))

printf( "\n The information has been updated!!!!\n\n" );

}

else if ( y\_n == 2 )

printf( "\n Return to Main Menu\n\n" );

else

printf( "\n Invalid Action\n\n" );

}

else

printf( "\n Invalid Customer ID\n\n" );

system( "pause" );

}

void sPurchase( memRec members[], shoeRec shoes[], int mcount, int scount )

{

int tmemid;

int tshoeid;

int discount;

int squan;

int i;

int ssize;

float shoeCost;

float shoeBill;

float disAmount, totalBill, finalBill;

int numshoes;

FILE \*billfile;

billfile = fopen( "bill.txt", "w" );

printTitle();

printf( " Enter Member ID (0 for Non-Member) : " );

scanf( "%d", &tmemid );

if (( tmemid >= 0 ) && ( tmemid <= mcount ))

{

if ( tmemid != 0 )

{

printTitle();

printf( " Member Id : %d\n\n", tmemid );

printf( " Member Name : %s %s\n\n", members[tmemid - 1].fname, members[tmemid - 1].lname );

system( "pause" );

}

shoeBill = 0;

numshoes = 0;

fprintf( billfile, " \*\*\*\*\*\* S H O E W E A R C O R N E R \*\*\*\*\*\*\n" );

fprintf( billfile, " #10 X Street, Penal\n\n" );

fprintf( billfile, " SHOE QUANTITY PRICE TOTAL PRICE\n" );

/\* Get all the shoes purchased to produce total shoe bill \*/

do

{

/\* Get the shoe id to purchase \*/

tshoeid = getShoe( shoes, scount );

if ( tshoeid != 0 )

{

do

{

/\* Print the three sizes available \*/

printTitle();

printf( " -----------------\n" );

printf( " | NO. | SHOE SIZE |\n" );

for ( i = 0; i < 3; i++ )

{

printf( " |-----|-----------|\n" );

printf( " | %2d | %2.1f |\n", i, shoes[tshoeid - 1].size[i] );

}

printf( " |-----|-----------|\n" );

printf( " | %2d | NONE |\n", i );

printf( " ------------------\n\n" );

printf( " Enter Shoe Size : " );

scanf( "%d", &ssize );

}

while (( ssize < 0 ) || ( ssize > 3 ));

printf( "\n Enter Quantity required (0 if none) : " );

scanf( "%d", &squan );

/\* There is a size that the user wants \*/

if (( squan != 0 ) && ( ssize != 0 ))

{

if ( squan <= shoes[tshoeid - 1].quantity[ssize] )

{

/\* Increase the number of shoes in this purchase \*/

numshoes = numshoes + squan;

shoeCost = squan \* shoes[tshoeid - 1].price;

shoeBill = shoeBill + shoeCost;

shoes[tshoeid - 1].quantity[ssize] = shoes[tshoeid - 1].quantity[ssize] - squan;

fprintf( billfile, "%s %2d $%6.2f $%6.2f\n", shoes[tshoeid - 1].shoeName, squan, shoes[tshoeid - 1].price, shoeCost );

}

else

{

printf( "\n Insufficient stock to fill order\n\n" );

system( "pause" );

}

}

else

printf( "\n No Size for the Customer\n\n" );

}

}

while ( tshoeid != 0 );

if ( shoeBill != 0.00 )

{

printTitle();

printf( " SHOE BILL - $%7.2f\n\n", shoeBill );

if ( tmemid != 0 )

{

discount = calcuDis( shoeBill );

disAmount = shoeBill \* discount/100;

finalBill = shoeBill - disAmount;

printf( " TOTAL BILL - $%7.2f\n\n", shoeBill );

printf( " DISCOUNT (%2d%) - ", discount );

printf( "$%7.2f\n\n", disAmount );

fprintf( billfile, " TOTAL BILL - $%7.2f\n\n", shoeBill );

fprintf( billfile, " DISCOUNT (%2d%) - ", discount );

fprintf( billfile, "$%7.2f\n\n", disAmount );

}

else

finalBill = shoeBill;

printf( " FINAL BILL - $%7.2f\n\n", finalBill );

fprintf( billfile, " FINAL BILL - $%7.2f\n\n", finalBill );

system( "pause" );

members[tmemid - 1].balance = members[tmemid - 1].balance + finalBill;

}

}

fclose( billfile );

}

void payment( memRec members[], int mcount )

{

int tmemid;

float pd, change;

int memloc;

printTitle();

printf( " Enter the Member ID : " );

scanf( "%d", &tmemid );

if (( tmemid > 0 ) && ( tmemid <= mcount ))

{

memloc = tmemid - 1;

printTitle();

printf( " MEMBER INFORMATION\n" );

printf( " ==================\n\n" );

printf( " MEMBER ID : %d\n", tmemid );

printf( " MEMBER NAME : %s %s\n", members[tmemid - 1].fname, members[tmemid - 1].lname );

printf( " BALANCE : $%7.2f\n\n", members[tmemid - 1].balance );

if ( members[memloc].balance == 0 )

printf( "\n No outstanding balance!!!\n\n" );

else

{

printf( " Enter Payment : $" );

scanf( "%f", &pd );

if ( pd < members[memloc].balance )

members[memloc].balance = members[memloc].balance - pd;

else if ( pd > members[tmemid - 1].balance )

{

change = pd - members[memloc].balance;

members[memloc].balance = 0.00;

printf( "\n Change : $%7.2f\n\n", change );

}

else

members[tmemid - 1].balance = 0.00;

printf( "\n NEW BALANCE : $%7.2f\n\n", members[memloc].balance );

}

}

system( "pause" );

}

int calcuDis( float shoeBill )

{

if ( shoeBill > 2000.00 )

return 20;

else

return 0;

}

int getShoe( shoeRec shoes[], int shoecount )

{

int i, tshoeid, shoet;

int shoeb;

int search, heading, tabcount;

char brand;

int numshoep;

FILE \*recfile;

/\* A file is created to store the product receipt to be printed \*/

recfile = fopen( "rec.txt", "w" );

fprintf( recfile, " \*\*\*\*\*\* S H O E W E A R C O R N E R \*\*\*\*\*\*\n" );

fprintf( recfile, " \*\*\* Shoes to Match Your Mood \*\*\*\n" );

fprintf( recfile, " #10 X Street, Penal\n\n" );

printTitle();

printf( " SEARCH CRITERIA\n" );

printf( " ===============\n\n" );

printf( " 1. SHOE TYPE\n\n" );

printf( " 2. SHOE BRAND\n\n" );

printf( " 3. ALL SHOES\n\n" );

printf( " 4. END SHOPPING\n\n\n" );

printf( " Enter the Search # : " );

scanf( "%d", &search );

if ( search == 1 )

shoet = getShoeType();

else if ( search == 2 )

shoeb = getShoeBrand();

else if ( search == 3 )

{

shoet = 0;

shoeb = 0;

}

else if ( search == 4 )

{

printf( "\n Your bill would be printed\n\n" );

system( "pause" );

tshoeid = 0;

}

else

{

printf( "\n Invalid Search Choice\n\n" );

system( "pause" );

}

printTitle();

if (( search >= 1 ) && ( search < 4 ))

{

/\* Indicates if any shoes purchased to not print empty list at the end \*/

numshoep = 0;

if ( search == 1 )

{

printTitle();

printf( " SHOE TYPE (" );

printShoeType( shoet );

printf( ") SEARCH\n\n" );

printf( " ---------------------------------------------------------------- \n" );

printf( "| SHOE | SHOE NAME | SHOE | GENDER | PRICE |\n" );

printf( "| ID | | BRAND | | |\n" );

printf( "|------|--------------------------|----------|--------|----------|\n" );

}

else if ( search == 2 )

{

printTitle();

printf( " SHOE BRAND (" );

printBrand( shoeb );

printf( ") SEARCH\n\n" );

printf( " ---------------------------------------------------------------- \n" );

printf( "| SHOE | SHOE NAME | SHOE | GENDER | PRICE |\n" );

printf( "| ID | | TYPE | | |\n" );

printf( "|------|--------------------------|----------|--------|----------|\n" );

}

else

{

printTitle();

printf( " ALL SHOES SEARCH\n\n" );

printf( "|\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\n" );

printf( " --------------------------------------------------------------------------- \n" );

printf( "| SHOE | SHOE NAME | SHOE | BRAND | GENDER | PRICE |\n" );

printf( "| ID | | TYPE | | | |\n" );

printf( "|------|--------------------------|----------|----------|--------|-----------|\n" );

}

for ( i = 0; i < shoecount; i++ )

{

if ( search == 1 )

{

if ( shoet == shoes[i].shoeType )

{

printf( "| %2d. | %25s", shoes[i].shoeID, shoes[i].shoeName );

printf( "| " );

printBrand( shoes[i].brand );

numshoep = numshoep + 1;

if ( shoes[i].gender == 1 )

printf( " | Female " );

else

printf( " | Male " );

printf( "| $%7.2f |\n", shoes[i].price );

fprintf( "| %2d. | %25s", shoes[i].shoeID, shoes[i].shoeName );

}

}

else if ( search == 2 )

{

if ( shoeb == shoes[i].brand )

{

printf( "| %2d. | %25s", shoes[i].shoeID, shoes[i].shoeName );

printf( "| " );

printShoeType( shoes[i].shoeType );

numshoep = numshoep + 1;

if ( shoes[i].gender == 1 )

printf( " | Female " );

else

printf( " | Male " );

printf( "| $%7.2f |\n", shoes[i].price );

}

}

else

{

printf( "| %2d. | %25s", shoes[i].shoeID, shoes[i].shoeName );

printf( "| " );

printBrand( shoes[i].brand );

printf( " | " );

printShoeType( shoes[i].shoeType );

/\* Checks the number of shoes printed for search \*/

numshoep = numshoep + 1;

if ( shoes[i].gender == 1 )

printf( " | Female " );

else

printf( " | Male " );

printf( "| $%8.2f |\n", shoes[i].price );

}

}/\* for \*/

if (( search == 1 ) || ( search == 2 ))

printf( "|\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\n" );

else

printf( "|\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_|\n" );

if ( numshoep == 0 )

{

printTitle();

printf( "\n\n NO SHOES AVAILABLE\n\n" );

system( "pause" );

}

else

{

printf( "\n\n Enter the Shoe ID to Purchase (0 to stop) : " );

scanf( "%d", &tshoeid );

}

}/\*if\*/

if ( search == 1 )

{

if ( shoes[tshoeid - 1].shoeType == shoet )

return tshoeid;

else

{

printf( "\n You must enter Shoe No for type selected\n\n" );

system( "pause" );

return 0;

}

}

else if ( search == 2 )

{

if ( shoes[tshoeid - 1].brand == shoeb )

return tshoeid;

else

{

printf( "\n You must enter Shoe No for Brand selected\n\n" );

system( "pause" );

return 0;

}

}

else if ( search == 3 )

return tshoeid;

else

return 0;

}

void shoeOptions( shoeRec shoes[], int shoecount )

{

int shoeopt;

do

{

printTitle();

printf( " \*\*\*\*\*\*\* SHOE OPTIONS \*\*\*\*\*\*\*\n\n" );

printf( " 1. Register New Shoe\n\n" );

printf( " 2. Update Shoe Information\n\n" );

printf( " 3. Return to Main Menu\n\n\n" );

printf( " Enter Shoe Option # : " );

scanf( "%d", &shoeopt );

if ( shoeopt == 1 )

shoeRegister( shoes, shoecount );

else if ( shoeopt == 2 )

shoeUpdate( shoes, shoecount );

else

{

printf( "\nReturn to Main Menu\n\n" );

system( "pause" );

}

}

while ( shoeopt != 3 );

}

void shoeRegister( shoeRec shoes[], int shoecount )

{

int i;

printTitle();

printf( " \*\*\*\*\*\* SHOE REGISTRATION \*\*\*\*\*\*\n\n" );

shoes[shoecount].shoeType = getShoeType();

shoes[shoecount].brand = getShoeBrand();

printf( "\n Enter Shoe Name : " );

fgets( shoes[shoecount].shoeName, 25, stdin );

fgets( shoes[shoecount].shoeName, 25, stdin );

shoes[shoecount].colour = getColor();

printTitle();

for ( i = 0; i < 3; i++ )

{

printf( "\n Enter Shoe Size : " );

scanf( "%f", &shoes[shoecount].size[i] );

printf( "\n Enter Quantity : " );

scanf( "%d", &shoes[shoecount].quantity[i] );

}

printf( "\n Enter Shoe Price : $" );

scanf( "%f", &shoes[shoecount].price );

printTitle();

printf( " NEW SHOE INFORMATION\n" );

printf( " ====================\n\n" );

printf( " SHOE ID - %d\n", shoecount );

printf( " SHOE TYPE - " );

printShoeType( shoes[shoecount].shoeType );

printf( "\n SHOE NAME - %s\n", shoes[shoecount].shoeName );

printf( " SHOE BRAND - " );

printBrand( shoes[shoecount].brand );

printf( "\n COLOUR - " );

printColor( shoes[shoecount].colour );

printf( "\n\n SHOE SIZE QUANTITY\n" );

for ( i = 0; i < 3; i++ )

{

printf( " %2.1f %d\n", shoes[shoecount].size[i], shoes[shoecount].quantity[i] );

}

printf( "\n PRICE - $%6.2f\n\n", shoes[shoecount].price );

printf( " The new shoe has been registered.\n" );

shoecount = shoecount + 1;

shoes[shoecount].shoeID = shoecount;

system( "pause" );

}

void shoeUpdate( shoeRec shoes[], int shoecount )

{

int choicechng;

int tshoeid;

int change;

int quan;

int i;

printTitle();

printf( " Enter Shoe ID : " );

scanf( "%d", &tshoeid );

if ( ( tshoeid > 0 ) && ( tshoeid < shoecount ))

{

printTitle();

printf( " SHOE INFORMATION\n" );

printf( " ================\n\n" );

printf( " SHOE ID - %d\n\n", tshoeid );

printf( " SHOE NAME - %s\n\n", shoes[tshoeid - 1].shoeName );

printf( " SHOE PRICE - $%6.2f\n\n", shoes[tshoeid - 1].price );

printf( " SHOE SIZE STOCK AMOUNT\n" );

for ( i = 0; i < 3; i++ )

printf( " %3.1f %d\n", shoes[tshoeid - 1].size[i], shoes[tshoeid - 1].quantity[i] );

printf( " 1. Quantity Size(%3.1f) %d\n", shoes[tshoeid - 1].size[0], shoes[tshoeid - 1].quantity[0] );

printf( " 2. Quantity Size(%3.1f) %d\n", shoes[tshoeid - 1].size[1], shoes[tshoeid - 1].quantity[1] );

printf( " 3. Quantity Size(%3.1f) %d\n", shoes[tshoeid - 1].size[2], shoes[tshoeid - 1].quantity[2] );

printf( " 4. Shoe Price\n" );

printf( " 5. No Change\n\n" );

printf( " Enter the information # to change : " );

scanf( "%d", &change );

if (( change == 1 ) || ( change == 2 ) || ( change == 3 ))

{

printf( " Enter the New Quantity : " );

scanf( "%d", &quan );

shoes[tshoeid - 1].quantity[change - 1] = shoes[tshoeid - 1].quantity[change - 1] + quan;

printTitle();

printf( " SHOE ID - %d\n\n", tshoeid );

printf( " SHOE NAME - %s\n\n", shoes[tshoeid - 1].shoeName );

printf( " SHOE PRICE - $%6.2f\n\n", shoes[tshoeid - 1].price );

printf( " SHOE SIZE STOCK AMOUNT\n" );

for ( i = 0; i < 5; i++ )

printf( " %3.1f %d\n", shoes[tshoeid - 1].size[i], shoes[tshoeid - 1].quantity[i] );

printf( "\n The shoe quantity has been successfully updated.\n" );

}

else if ( change == 4 )

{

printf( "\n Enter the New Shoe Price : $" );

scanf( "%f", &shoes[tshoeid - 1].price );

printf( "\n The shoe price has been successfully updated.\n" );

}

else

printf( "\n Thanks for viewing information \n" );

}

else

printf( "\n Invalid choice, only change price and quantity\n" );

system( "pause" );

}

void busiOption( shoeRec shoes[], int shoecount )

{

int choice;

int brand;

do

{

printTitle();

printf( " \*\*\*\*\*\* BUSINESS OPTIONS \*\*\*\*\*\*\n\n" );

printf( "1. About Business\n\n" );

printf( "2. Display Brands\n\n" );

printf( "3. Return to Main Menu\n\n\n" );

printf( "Please enter option : " );

scanf( "%d", &choice );

if ( choice == 1 )

busiAbt();

else if ( choice == 2 )

brand = getShoeBrand();

else if ( choice == 3 )

{

printf( "\nReturning to Main Menu\n\n" );

system( "pause" );

}

else

{

printf( "\nInvalid Option\n\n" );

system( "pause" );

}

}

while ( choice != 3 );

}

void busiAbt()

{

int linecount;

char infoline[100];

FILE \*abtfile;

abtfile = fopen( "about.txt", "r" );

if ( !abtfile )

printf( "Cannot open file" );

else

{

printTitle();

linecount = 1;

while ( fgets( infoline, sizeof(infoline), abtfile ))

{

printf( "%s", infoline );

if (( linecount % 5 == 0 ))

{

printf("\n");

system( "pause" );

printTitle();

}

linecount++;

}

}

system( "pause" );

}

int getShoeType()

{

int shoetype;

printTitle();

printf( " SHOE CATEGORIES\n" );

printf( " ===============\n\n" );

printf( " 1. Athletic Shoes and Sneakers\n" );

printf( " 2. Heels & Pumps\n" );

printf( " 3. Wedges\n" );

printf( " 4. Designer Shoes\n" );

printf( " 5. Sandals\n" );

printf( " 6. Slippers\n" );

printf( " 7. Slides & Mules\n" );

printf( " 8. Dress Shoes\n" );

printf( " 9. Boots\n" );

printf( " 10. NEW ARRIVALS\n\n" );

printf( " Please enter Shoe Category # : " );

scanf( "%d", &shoetype );

return shoetype;

}

void printShoeType( int shoetype )

{

if ( shoetype == 1 )

printf( "Sneakers" );

else if ( shoetype == 2 )

printf( "Heels " );

else if ( shoetype == 3 )

printf( "Wedges " );

else if ( shoetype == 4 )

printf( "Designer" );

else if ( shoetype == 5 )

printf( "Sandals " );

else if ( shoetype == 6 )

printf( "Slippers" );

else if ( shoetype == 7 )

printf( "Slides " );

else if ( shoetype == 8 )

printf( "Dress " );

else if ( shoetype == 9 )

printf( "Boots " );

else

printf( "New " );

}

int getShoeBrand()

{

int brand;

int brandNum;

do

{

printTitle();

printf( " BRANDS AVAILABLE\n" );

printf( " ================\n\n" );

printf( " -------------------------- \n" );

printf( " | BRAND | BRAND NAME |\n" );

printf( " |-------|------------------|\n" );

printf( " | 1. | Adidas |\n" );

printf( " | 2. | Nike |\n" );

printf( " | 3. | Reebok |\n" );

printf( " | 4. | Puma |\n" );

printf( " | 5. | Gucci |\n" );

printf( " | 6. | Fila |\n" );

printf( " | 7. | Louis Vuitton |\n" );

printf( " | 8. | Champion |\n" );

printf( " | 9. | Clarks |\n" );

printf( " | 10. | Fioni |\n" );

printf( " -------------------------- \n" );

printf( " 0. NO BRAND OR VIEWED BRANDS\n" );

printf( " Enter Brand # : " );

scanf( "%d", &brand );

}

while (( brand < 0 ) || ( brand > 10 ));

return brand;

}

void printBrand( int brand )

{

char wbrand;

if ( brand == 1 )

printf( "Adidas " );

else if ( brand == 2 )

printf( "Nike " );

else if ( brand == 3 )

printf( "Reebok " );

else if ( brand == 4 )

printf( "Puma " );

else if ( brand == 5 )

printf( "Gucci " );

else if ( brand == 6 )

printf( "Fila " );

else if ( brand == 7 )

printf( "Madden " );

else if ( brand == 8 )

printf( "Champion" );

else if ( brand == 9 )

printf( "Clarks " );

else

printf( "Fioni " );

}

## Test Plan

Member

|  |  |  |  |
| --- | --- | --- | --- |
| Type of Data | Test Data | Expected Results | Actual Results |
| Normal |  |  |  |
| Normal |  |  |  |
| Normal |  |  |  |
| Normal |  |  |  |
| Normal |  |  |  |
| Extreme |  |  |  |
| Extreme |  |  |  |
| Extreme |  |  |  |
| Extraneous |  |  |  |
| Extraneous |  |  |  |

Shoe

Business