```c
/******************************************************************************************************************************************
This is to certify that this project is my own work, based on my personal efforts in studying and applying the concepts learned.
I have constructed the functions and their respective algorithms and corresponding code by myself. The program was run, tested,
and debugged by my own efforts. I further certify that I have not copied in part or whole or otherwise plagiarized the work of
other students and/or persons.

                    AGULTO, JULIANA MARIE B. ,          DLSU ID# 11927097
******************************************************************************************************************************************/

#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<conio.h>

#define MAX10 11 // string of at most 10 characters
#define MAX15 18 // string of at most 15 characters
#define MAX20 23 // string of at most 20 characters
#define MAX30 33 // string of at most 30 characters

typedef char string10[MAX10];
typedef char string15[MAX15];
typedef char string20[MAX20];
typedef char string30[MAX30];

// this structure is for the user information
typedef struct UserTag
```

```
{
        string10 Password;      // user password
        int ID;                         // user id
        string15 ContNumber;    // user contact number
        string20 Name;          // user name
        string30 Address;       // user address

} UserInfo;
// this structure is for the item information
typedef struct ItemTag
{
        float iPrice;           // item price
        int iQty;                       // item quantity
        int iID;                // item id
        string15 iCategory;     // item category
        string20 iName;         // item name
        string30 iDescription;          // item description
        int sID;                // seller id

} ItemInfo;

// this structure is for the date information
typedef struct DateTag
{
        int nMonth;
        int nDay;
        int nYear;

} DateInfo;

// this structure is for the cart information
typedef struct CartTag
{
        int cQty;                       // quantity in cart
        ItemInfo cItem;         // cart item = structure of ItemInfo
        float cPrice;           // cart price of the item

} CartInfo;

// this structure is for the transaction information
typedef struct TransacTag
{
        DateInfo tD;    // transaction date
        int tIndex;             // transaction index
        int tQty[5];    // quantity of the item to be checked out
        float tPrice;   //total price
        //item info
        int iId[5];                     // item id
        float iPrice[5];  // item price
        string20 iName[5];      // item name
        float iDiscount[5];     // item discount
        //seller info
```

```
	int sId;			// seller id
	string20 sName;			// seller name
	string30 sAddress;		// seller address
	//buyer info
	int bId;			// buyer id
	string20 bName;			// buyer name
	string30 bAddress;		// buyer address

} TransacInfo;

// this structure is for the discount information
typedef struct DiscountTag
{
	int sID;		// seller id
	int iID;		// item id
	float Discount;	// discount entered

}	DiscountInfo;

//Preprocessor directives ------------------
void saveUsers(UserInfo *aUserData, int CountIndex);
void openUsers(UserInfo *aUserData, int *CountIndex);
void saveItems(ItemInfo *aItemData, int nSIndex);
void openItems(ItemInfo* aItemData, int *CountIndex);
void saveCart(CartInfo *aCartData, int nCartIndex, int ID);
void openCart(CartInfo *aCartData, int *nCartIndex, int ID);
void saveTransac(TransacInfo TransacData);
void openTransac(TransacInfo *TransacData, int *nTransacIndex);

char MainMenu();
char UserMenu();
char SellMenu();
char editStock();
char BuyMenu();
char showTransacMenu();
char editCart();
char checkOut();
char AdminMenu();
char Confirm();

int checkUserID(UserInfo *aUserData, int nCountIndex, int checkID);
int checkProductID(ItemInfo *aItemData, int nSIndex, int checkID);
int checkCSellerID(CartInfo *aCartData, int nCartIndex, int checkID);
int checkCProductID(CartInfo *aCartData, int nCartIndex, int checkID);
int checkContact(char *Contact);

UserInfo Register(UserInfo *aUserData, int *nCountIndex);
void displayUser(UserInfo aUserData);
void sortID(UserInfo *aUserData, int nCountIndex);
int Log_In(UserInfo *aUserData, int nCountIndex, int *uID);

ItemInfo addItem(ItemInfo *aItemData, int *nSIndex, int sID);
```

```
void displayProduct(ItemInfo ItemData);

int sellBag20(ItemInfo *aItemData, int nSIndex, ItemInfo *aItem20, int ID);

void showProducts(ItemInfo *aItemData, int nSIndex, int nCheck, int pID);

void sortProducts(ItemInfo *aItemData, int nSIndex);

void Replenish_Reduce(ItemInfo *aItem20, int nIndex20, ItemInfo *aItemData, int nSIndex, int pID, int nCheck);

void changePrice(ItemInfo *aItem20, int nIndex20, ItemInfo *aItemData, int nSIndex, int pID);

void changeName(ItemInfo *aItem20, int nIndex20, ItemInfo *aItemData, int nSIndex, int pID);

void changeCategory(ItemInfo *aItem20, int nIndex20, ItemInfo *aItemData, int nSIndex, int pID);

void changeDescription(ItemInfo *aItem20, int nIndex20, ItemInfo *aItemData, int nSIndex, int pID);

void showLowProducts(ItemInfo *aItemData, int nSIndex);

void addDiscount(ItemInfo *aItemData, int nSIndex, int ID);

void removeDiscount(ItemInfo *aItemData, int nSIndex, int ID);

float openDiscount(int pID, int sID);


int enterPID(ItemInfo *aItemData, int nSIndex);

void allProducts(ItemInfo *aItemData, int nSIndex, int uID, UserInfo *aUserData, int nCountIndex);

void viewDiscount(ItemInfo *aItemData, int nSIndex, int uID, UserInfo *aUserData, int nCountIndex);

void specificSeller(ItemInfo *aItemData, int nSIndex, int uID);

void searchCategory(ItemInfo *aItemData, int nSIndex, int uID);

void searchName(ItemInfo *aItemData, int nSIndex, int uID);

void displayCart(CartInfo *aCartData, int nCartIndex);


CartInfo addCart(CartInfo *aCartData, int *nCartIndex, ItemInfo *aItemData, int nSIndex, int bID);

void compareItem(CartInfo *aCartData, int nCartIndex, ItemInfo *aItemData, int nSIndex);

float roundUp(float fNum) ;

void removeSeller(CartInfo *aCartData, int *nCartIndex);
void removeItem(CartInfo *aCartData, int *nCartIndex, int pID);
void editQty(CartInfo *aCartData, int *nCartIndex, ItemInfo *aItemData, int nSIndex);

void getDate(int *nMonth, int *nDay, int *nYear);
TransacInfo confirmDate(TransacInfo TransacData);
void sortDate(TransacInfo *TransacData, int nTransacIndex);
TransacInfo transacSeller(CartInfo *aCartTemp, int nCartTemp, ItemInfo *aItemData, int nSIndex, TransacInfo Transac, int sID);
TransacInfo transacItem(CartInfo *aCartData, int nCartIndex, ItemInfo *aItemData, int nSIndex, TransacInfo TransacData);
TransacInfo completeInfo(UserInfo *UserData, int nCountIndex, TransacInfo Transac, int nID);
void displayReceipt(TransacInfo Transac);

void adminUsers(UserInfo *aUserData, int nCountIndex);
void adminSellers(ItemInfo *aItemData, int nSIndex, UserInfo *aUserData, int nCountIndex);
void adminTotalSales();
int TransacSID(UserInfo *aUserData, TransacInfo *TransacData, int nTransacIndex);
void adminSellerSales(int nID, string20 UserName);
void adminShopaholics(UserInfo *aUserData, int nCountIndex);
void userReceipt(int ID);
//-----------------
/* This function is for saving the array list of user information to the text file
[ @param        (UserInfo) *aUserData = array list of users                     ]
[                      (int) nCountIndex = index of array list of users    ]
[ @return       no return value                                                                    ] */
void saveUsers(UserInfo *aUserData, int nCountIndex)
{
        int i;
```

```
	FILE *pFile;

	pFile = fopen("Users.txt", "w");

	for(i = 0; i < nCountIndex; i++)
	{
		fprintf(pFile, "%d %s\n", aUserData[i].ID, aUserData[i].Password);
		fprintf(pFile, "%s\n", aUserData[i].Name);
		fprintf(pFile, "%s\n", aUserData[i].Address);
		fprintf(pFile, "%s\n\n", aUserData[i].ContNumber);
	}
	fclose(pFile);
}
/* This function is for opening the user information from the text file
[ @param	(UserInfo) *aUserData = array list of users						]
[		(int) *nCountIndex = index of array list of users					]
[ @return	(UserInfo) *aUserData = returns the user information to the array of users	]
		(int) *nCountIndex = returns the index of the user of the text file		] */
void openUsers(UserInfo *aUserData, int *nCountIndex)
{
	int i = 0;
	char c;
	FILE *pFile;

	if ((pFile = fopen("Users.txt","r")) == NULL)
      printf("\n\t\t\t\t\t  Xx NO PREVIOUS USERS IN TEXT FILE xX\n");
		else
		{
			while(fscanf(pFile, "%d", &aUserData[i].ID)!= EOF)
			{
				c = fgetc(pFile); // gets the characters in the text file
				fgets(aUserData[i].Password, MAX10+1, pFile);
				aUserData[i].Password[strlen(aUserData[i].Password) - 1] = '\0';

				fgets(aUserData[i].Name, MAX20, pFile);
				aUserData[i].Name[strlen(aUserData[i].Name) - 1] = '\0';

				fgets(aUserData[i].Address, MAX30, pFile);;
				aUserData[i].Address[strlen(aUserData[i].Address) - 1] = '\0';

				fscanf(pFile, "%s", aUserData[i].ContNumber);
				i++;
			}
			*nCountIndex = i;
	}
	fclose(pFile);
}
/* This function is for saving the array list of item information to the text file
[ @param	(ItemInfo) *aItemData = array list of items			]
[			(int) nSIndex = index of array list of items		]
[ @return	no return value								] */
void saveItems(ItemInfo *aItemData, int nSIndex)
```

```
{
	int i;
	FILE *pFile;

	pFile = fopen("Items.txt", "w"); // dapat hindi specific na directory

	for(i = 0; i < nSIndex; i++)
	{
		fprintf(pFile, "%d %d\n", aItemData[i].iID, aItemData[i].sID);
		fprintf(pFile, "%s\n", aItemData[i].iName);
		fprintf(pFile, "%s\n", aItemData[i].iCategory);
		fprintf(pFile, "%s\n", aItemData[i].iDescription);
		fprintf(pFile, "%d %.2f\n\n", aItemData[i].iQty, aItemData[i].iPrice);
	}
	fclose(pFile);
}
/* This function is for opening the item information from the text file
[ @param	(ItemInfo) *aItemData = array list of items								]
[		(int) nSIndex = index of array list of items								]
[ @return	(ItemInfo) *aItemData = returns the item information to the array of items	]
[		(int) nSIndex = returns the index of the items of the text file				] */
void openItems(ItemInfo *aItemData, int *nSIndex)
{
	int i = 0;
	char c;
	FILE *pFile;
```

```
            if ((pFile = fopen("Items.txt","r")) == NULL)
                        printf("\n\t\t\t\t\t\t  Xx NO PREVIOUS ITEMS IN TEXT FILE xX\n");
            else
            {
                        while(fscanf(pFile, "%d", &aItemData[i].iID)!= EOF)
                        {
                                    fscanf(pFile, "%d", &aItemData[i].sID);
                                    c = fgetc(pFile);
                                    fgets(aItemData[i].iName, MAX20, pFile);
                                    aItemData[i].iName[strlen(aItemData[i].iName) - 1] = '\0';
                                    fgets(aItemData[i].iCategory, MAX15, pFile);
                                    aItemData[i].iCategory[strlen(aItemData[i].iCategory) - 1] = '\0';
                                    fgets(aItemData[i].iDescription, MAX30, pFile);
                                    aItemData[i].iDescription[strlen(aItemData[i].iDescription) - 1] = '\0';
                                    fscanf(pFile, "%d", &aItemData[i].iQty);
                                    fscanf(pFile, "%f", &aItemData[i].iPrice);
                                    i++;
                        }
                        *nSIndex = i;
  }
  fclose(pFile);
}
/* This function is for saving the array list of cart items information to the binary file
[ @param          (CartInfo) *aCartData = array list of items from the users' cart                    ]
[                                       (int) nCartIndex = index of array list of items from the users' cart ]
[                                   (int) ID = user id                                                                                                                          ]
[ @return          no return value                                                                                                                                              ] */
void saveCart(CartInfo *aCartData, int nCartIndex, int ID)
{
            int i;
            FILE *pFile;
            string15 sID;

            itoa(ID, sID, 10);// itoa converts (int) to string
            pFile = fopen (strcat(sID,".bag"), "wb");

            for(i = 0; i < nCartIndex; i++)
                        fwrite(&aCartData[i], sizeof(struct CartTag), 1, pFile);

            fclose(pFile);
}
/* This function is for opening the cart items information from the binary file
[ @param           (CartInfo) *aCartData = array list of items from the users' cart                                                  ]
[                                   (int) *nCartIndex = index of array list of items from the users' cart                                          ]
[                                   (int) ID = user id                                                                                                                                  ]
[ @return          (int) *nCartIndex = returns the index of the cart items of the user of the text file  ] */
void openCart(CartInfo *aCartData, int *nCartIndex, int ID)
{
            int i;
            FILE *pFile;
            string15 sID;
```

```
	itoa(ID, sID, 10);// itoa converts (int) to string
	pFile = fopen (strcat(sID,".bag"), "rb");
	if(pFile)
	{
		for(i = 0; !feof(pFile); i++)
			fread(&aCartData[i], sizeof(struct CartTag), 1, pFile);

		i--;	// since there is a new line it copies the information before it and repeats the same information
		*nCartIndex = i;
	}
	fclose(pFile);
}
/* This function is for saving the transaced information to the binary file
[ @param	(TransacInfo) TransacData = a single structure of the transaction information	]
[ @return	no return value									]	*/
void saveTransac(TransacInfo TransacData)
{
	FILE *pFile;
	pFile = fopen("Transactions.dat","ab");
	fwrite(&TransacData, sizeof(struct TransacTag), 1, pFile);
	fclose(pFile);
}
/* This function is for opening the transaced information from the binary file
[ @param	(TransacInfo) *aTransacData = array list of transaced items			]
[ @param	(int) *nTransacIndex = index of array list of transaced items			]
[ @return	(int) *nTransacIndex = returns the index of the transaced items of the user of the binary file	]	*/
```

```
void openTransac(TransacInfo *aTransacData, int *nTransacIndex)
{
	int i, nTempIndex = 0;
	int nSMonth, nSDay, nSYear, nEMonth, nEDay, nEYear;
	FILE *pFile;
	TransacInfo tempTransac[500];  // temporarily stores all of the transactions

	printf("\n\t\t\t\t\t Enter Start Date\n");
	getDate(&nSMonth, &nSDay, &nSYear);

	do
	{
		printf("\n\t\t\t\t\t Enter End Date\n");
		getDate(&nEMonth, &nEDay, &nEYear);
		if(nSMonth * 100 + nSDay + nSYear * 10000 > nEMonth * 100 + nEDay + nEYear * 10000)
			printf("\n\t\t\t\t\t\t\tXx INVALID END DATE xX\n");
	}while(nSMonth * 100 + nSDay + nSYear * 10000 > nEMonth * 100 + nEDay + nEYear * 10000);    // makes sure that the end date is greater than or equal to start date

	pFile = fopen("Transactions.dat","rb");
	if(!pFile)
		printf("\n\t\t\t\t\t\t Xx NO PREVIOUS ITEMS IN BINARY FILE xX\n");
	else
	{
		for(nTempIndex = 0; !feof(pFile) && nTempIndex < 500; nTempIndex++) // copies all the transactions to the temporary array
			fread(&tempTransac[nTempIndex], sizeof(struct TransacTag), 1, pFile);
		nTempIndex--;
```

```
			for(i = 0; i < nTempIndex; i++)     // this will loop the temporary array
			{
				if(nSDay + nSMonth * 100 + nSYear * 10000 <= tempTransac[i].tD.nDay +  tempTransac[i].tD.nMonth * 100 + tempTransac[i].tD.nYear * 10000 && nEDay + nEMonth * 100 +
nEYear * 10000 >= tempTransac[i].tD.nDay +  tempTransac[i].tD.nMonth * 100 + tempTransac[i].tD.nYear * 10000)
				{
					aTransacData[*nTransacIndex] = tempTransac[i];		// copies the structure that falls between the date to the aTransacData
					*nTransacIndex = *nTransacIndex + 1;
				}
			}
			if(*nTransacIndex < 1)    // if no transaction falls in the start and end date
				printf("\n\n\t\t\t\t\tXx SORRY, NO TRANSACTION FOUND IN ENTERED DATE xX\n");
			else
				printf("\n\n\t\t\t\t\t\tDate ::  %d / %d / %d - %d / %d / %d", nSMonth, nSDay, nSYear, nEMonth, nEDay, nEYear);
		}
		fclose(pFile);
}
/* This function is for the main menu selection
[ @param		no input parameter											]
[ @return		(char) cChoice = returns the entered character by the user		]		*/
char MainMenu()
{
		char cChoice;
		fflush(stdin);
		printf("\n\t\t\t\t\t+ - - - - - - - M A I N   M E N U - - - - - - - - +\n");
		printf("\t\t\t\t\t|						|\n");
		printf("\t\t\t\t\t|  [1] Register					|\n");
```

```
	printf("\t\t\t\t\t|  [2] User Menu                              |\n");
	printf("\t\t\t\t\t|  [3] Admin                                  |\n");
	printf("\t\t\t\t\t|  [0] Exit                                   |\n");
	printf("\t\t\t\t\t|                                             |\n");
	printf("\t\t\t\t\t+ - - - - - - - - - - - - - - - - - - - - - - - - - +\n");
	printf("\n\t\t\t\t\t Enter Choice :: ") ;
	scanf(" %c", &cChoice);
	return cChoice;
}
/* This function is for the user menu selection
[ @param	no input parameter										]
[ @return	(char) cChoice = returns the entered character by the user	]		*/
char UserMenu()
{
	char cChoice;
	fflush(stdin);
	printf("\t\t\t\t\t|                                             |\n");
	printf("\t\t\t\t\t|  [1] Sell Menu                              |\n");
	printf("\t\t\t\t\t|  [2] Buy Menu                               |\n");
	printf("\t\t\t\t\t|  [3] Show Transactions                      |\n");	// new feature
	printf("\t\t\t\t\t|  [0] Exit                                   |\n");
	printf("\t\t\t\t\t|                                             |\n");
	printf("\t\t\t\t\t+ - - - - - - - - - - - - - - - - - - - - - - - - - +\n");
	printf("\n\t\t\t\t\t Enter Choice :: ") ;
	scanf(" %c", &cChoice);
	return cChoice;
```

```
}
/* This function is for the sell menu selection
[ @param          no input parameter                                                                                      ]
[ @return         (char) cChoice = returns the entered character by the user          ]           */
char SellMenu()
{
        char cChoice;
        fflush(stdin);
        printf("\t\t\t\t\t|                                    |\n");
        printf("\t\t\t\t\t|  [1] Add New Item                  |\n");
        printf("\t\t\t\t\t|  [2] Edit Stock                    |\n");
        printf("\t\t\t\t\t|  [3] Show My Products              |\n");
        printf("\t\t\t\t\t|  [4] Show My Low Stock Products    |\n");
        printf("\t\t\t\t\t|  [5] Add Discount                  |\n");       // new feature
        printf("\t\t\t\t\t|  [6] Remove Discount               |\n");       // new feature
        printf("\t\t\t\t\t|  [0] Exit                          |\n");
        printf("\t\t\t\t\t|                                    |\n");
        printf("\t\t\t\t\t+ - - - - - - - - - - - - - - - - - - - - - - - - +\n");
        printf("\n\t\t\t\t\t Choice :: ");
        scanf(" %c", &cChoice);

        return cChoice;
}
/* This function is for the edit stock selection
[ @param          no input parameter                                                                                      ]
[ @return         (char) cChoice = returns the entered character by the user          ]           */
```

```
char editStock()
{
	char cChoice;
	fflush(stdin);
	printf("\n\t\t\t\t\t+ - - - - - - - E D I T   S T O C K - - - - - - - +\n");
	printf("\t\t\t\t\t|                                                 |\n");
	printf("\t\t\t\t\t|  [1] Replenish                                  |\n");
	printf("\t\t\t\t\t|  [2] Reduce Quantity                            |\n");	// new feature
	printf("\t\t\t\t\t|  [3] Change Price                               |\n");
	printf("\t\t\t\t\t|  [4] Change Item Name                           |\n");
	printf("\t\t\t\t\t|  [5] Change Category                            |\n");
	printf("\t\t\t\t\t|  [6] Change Description                         |\n");
	printf("\t\t\t\t\t|  [0] Finish Editing                             |\n");
	printf("\t\t\t\t\t|                                                 |\n");
	printf("\t\t\t\t\t+ - - - - - - - - - - - - - - - - - - - - - - - - +\n");
	printf("\n\t\t\t\t\t Choice :: ");
	scanf(" %c", &cChoice);
	return cChoice;
}
/* This function is for the buy menu selection
[ @param	no input parameter	]
[ @return	(char) cChoice = returns the entered character by the user	]	*/
char BuyMenu()
{
	char cChoice;
	fflush(stdin);
	printf("\t\t\t\t\t|                                          |\n");
	printf("\t\t\t\t\t|  [1] View All Products                   |\n");
	printf("\t\t\t\t\t|  [2] View Discounted Products            |\n");// new feature
	printf("\t\t\t\t\t|  [3] Show All Products by Specific Seller   |\n");
	printf("\t\t\t\t\t|  [4] Search Products by Category         |\n");
	printf("\t\t\t\t\t|  [5] Search Products by Name             |\n");
	printf("\t\t\t\t\t|  [6] Add to Cart                         |\n");
	printf("\t\t\t\t\t|  [7] Edit Cart                           |\n");
	printf("\t\t\t\t\t|  [8] Check Out                           |\n");
	printf("\t\t\t\t\t|  [9] Display / Compare Items             |\n");  // new feature
	printf("\t\t\t\t\t|  [0] Exit                                |\n");
	printf("\t\t\t\t\t|                                          |\n");
	printf("\t\t\t\t\t+ - - - - - - - - - - - - - - - - - - - - - - - - +\n");
	printf("\n\t\t\t\t\t Choice :: ");
	scanf(" %c", &cChoice);
	return cChoice;
}
/* This function is for the transacaction menu selection
[ @param	no input parameter								]
[ @return	(char) cChoice = returns the entered character by the user	]	*/
char showTransacMenu()
{
	char cChoice;
	fflush(stdin);
	printf("\t\t\t\t\t|                                          |\n");
	printf("\t\t\t\t\t|  [1] Items Sold                          |\n");
	printf("\t\t\t\t\t|  [2] Items Bought                    |\n");
	printf("\t\t\t\t\t|  [0] Exit                            |\n");
	printf("\t\t\t\t\t|                                      |\n");
	printf("\t\t\t\t\t+ - - - - - - - - - - - - - - - - - - - - - - - - +\n");
	printf("\n\t\t\t\t\t Choice :: ");
	scanf(" %c", &cChoice);

	return cChoice;
}
/* This function is for the edit cart selection
[ @param	no input parameter									]
[ @return	(char) cChoice = returns the entered character by the user	]	*/
char editCart()
{
	char cChoice;
	fflush(stdin);
	printf("\t\t\t\t\t|                                      |\n");
	printf("\t\t\t\t\t|  [1] Remove All Items From Seller    |\n");
	printf("\t\t\t\t\t|  [2] Remove Specific Item            |\n");
	printf("\t\t\t\t\t|  [3] Edit Quantity                   |\n");
	printf("\t\t\t\t\t|  [0] Finish Edit Cart                |\n");
	printf("\t\t\t\t\t|                                      |\n");
	printf("\t\t\t\t\t+ - - - - - - - - - - - - - - - - - - - - - - - - +\n");
	printf("\n\t\t\t\t\t Choice :: ");
	scanf(" %c", &cChoice);
	return cChoice;
```

```
}
/* This function is for the checkout selection
[ @param        no input parameter                                                          ]
[ @return       (char) cChoice = returns the entered character by the user      ]       */
char checkOut()
{
        char cChoice;
        printf("\t\t\t\t\t|                         |\n");
        printf("\t\t\t\t\t|  [1] All                |\n");
        printf("\t\t\t\t\t|  [2] By a Specific Seller |\n");
        printf("\t\t\t\t\t|  [3] Specific Item      |\n");
        printf("\t\t\t\t\t|  [0] Exit               |\n");
        printf("\t\t\t\t\t|                         |\n");
        printf("\t\t\t\t\t+ - - - - - - - - - - - - - - - - - - - - - - - - - +\n");
        printf("\n\t\t\t\t\tChoice :: ");
        scanf(" %c", &cChoice);

        return cChoice;
}
/* This function is for the admin menu selection
[ @param        no input parameter                                                          ]
[ @return       (char) cChoice = returns the entered character by the user      ]       */
char AdminMenu()
{
        char cChoice;
        fflush(stdin);
```

```
	printf("\t\t\t\t\t|                                             |\n");
	printf("\t\t\t\t\t|  [1] Show All Users                          |\n");
	printf("\t\t\t\t\t|  [2] Show All Sellers                        |\n");
	printf("\t\t\t\t\t|  [3] Show Total Sales in Given Duration      |\n");
	printf("\t\t\t\t\t|  [4] Show Seller Sales                       |\n");
	printf("\t\t\t\t\t|  [5] Show Shopaholics                        |\n");
	printf("\t\t\t\t\t|  [6] Show All Transactions by Specific Seller  |\n");	// new feature
	printf("\t\t\t\t\t|  [7] Show All Transactions by Specific Buyer   |\n");	// new feature
	printf("\t\t\t\t\t|  [0] Exit                                    |\n");
	printf("\t\t\t\t\t|                                             |\n");
	printf("\t\t\t\t\t+ - - - - - - - - - - - - - - - - - - - - - - - - - +\n");
	printf("\n\t\t\t\t\t Choice :: ");
	scanf(" %c", &cChoice);
	return cChoice;
}
/* This function is for the confirm, re-enter, cancel selection
[ @param	no input parameter								]
[ @return	(char) cChoice = returns the entered character by the user	]	*/
char Confirm()
{
	char cCRC;
	int X = 1;
	do
	{
		fflush(stdin);
		printf("\n\t\t\t\t\t\t\t<<>><<>><<<>><<>><<>>\n");
		printf("\t\t\t\t\t\t\t<<                >>\n");
		printf("\t\t\t\t\t\t\t<< [1] Confirm     >>\n");
		printf("\t\t\t\t\t\t\t<< [2] Re- Enter   >>\n");
		printf("\t\t\t\t\t\t\t<< [0] Cancel      >>\n");
		printf("\t\t\t\t\t\t\t<<                >>\n");
		printf("\t\t\t\t\t\t\t<<>><<>><<<>><<>><<>>\n");
		printf("\n\t\t\t\t\t\t\t Choice :: ");
		scanf(" %c", &cCRC);

		if (cCRC == '0' || cCRC == '1' ||cCRC == '2')
			X = 0;
		else
			printf("\n\t\t\t\t\t\t\tXx INVALID CHOICE xX\n");
	}while(X);
	return cCRC;
}
/* This function is for checking the user id from the list of users
[	@param		(UserInfo) *aUserData = array list of users						]
[	@param		(int) nCountIndex = index of array list of users				]
[	@param		(int) checkID = user id											]
[	@return		(int) nCheck = returns 1 if user is found and -1 if not		]	*/
int checkUserID(UserInfo *aUserData, int nCountIndex, int checkID)
{
	int i, nCheck = -1;
	for(i = 0; i < nCountIndex; i++)
		if(checkID == aUserData[i].ID)
```

```
				nCheck = 1; // if found
		return nCheck;
}
/* This function is for checking the product id from the list of items
[		@param		(ItemInfo) *aItemData = array list of items				]
[		@param		(int) nSIndex = index of array list of items			]
[		@param		(int) checkID = product id						]
[		@return		(int) nCheck = returns 1 if user is found and -1 if not	]		*/
int checkProductID(ItemInfo *aItemData, int nSIndex, int checkID)
{
		int i, nCheck = -1;

		for(i=0; i < nSIndex; i++)
		{
			if(checkID == aItemData[i].iID)
				nCheck = 1; // if found
		}
		return nCheck;
}
/* This function is for checking the seller id from the cart
[		@param		(CartInfo) *aCartData = array list of items from the cart		]
[		@param		(int) nCartIndex = index of array list of items from the cart		]
[		@param		(int) checkID = seller id							]
[		@return		(int) nCheck = returns 1 if user is found and 0 if not			]		*/
int checkCSellerID(CartInfo *aCartData, int nCartIndex, int checkID)
{
	int i, nCheck = 0;

	for(i = 0; i < nCartIndex; i++)
	{
		if(checkID == aCartData[i].cItem.sID)
			nCheck = 1; // if found
	}
	return nCheck;
}
/* This function is for checking the product id from the cart
[	@param		(CartInfo) *aCartData = array list of items from the cart		]
[	@param		(int) nCartIndex = index of array list of items from the cart		]
[	@param		(int) checkID = seller id								]
[	@return		(int) nCheck = returns 1 if user is found and 0 if not				]	*/
int checkCProductID(CartInfo *aCartData, int nCartIndex, int checkID)
{
	int i, nCheck = 0;

	for(i = 0; i < nCartIndex; i++)
	{
		if(checkID == aCartData[i].cItem.iID)
			nCheck = 1; // if found
	}
	return nCheck;
}
/* This function is for checking the contact number if a letter exists
```

```
[		@param		(char) *Contact = array used for the contact number					]
[		@return		(int) nCheck = returns 1 if user is found and -1 if not				]		*/
int checkContact(char *Contact)
{
		int i, nCheck = -1;

		for(i = 0; i < strlen(Contact); i++)
				if((Contact[i] >= 'A' && Contact[i] <= 'Z') || (Contact[i] >= 'a' && Contact[i] <= 'z'))
						nCheck = 1;
		if (nCheck == 1)
				printf("\n\t\t\t\t\t\tXx CONTACT NUMBER SHOULD BE NUMERICAL xX\n\n");
		return nCheck;
}

/* This function is for registering a new user and input the needed informations
[		@param		(UserInfo) *aUserData = array list of users										]
[					(int) *nCountIndex = index of array list of users									]
[		@return		(UserInfo) User = returns the entered information to the array of users						]
[					(int) *nCountIndex = will increment the index if the user confirmed the entered information	]		*/
UserInfo Register(UserInfo *aUserData, int *nCountIndex)
{
		UserInfo User;
		char cChoice;
		int nInput;

		do
```

```
{
	fflush(stdin);
	do
	{
		//user id
		printf("\t\t\t\t\t ID :: ");
		nInput = scanf("%d", &User.ID);
		fflush(stdin);

		if(checkUserID(aUserData, *nCountIndex, User.ID) == 1)
			printf("\t\t\t\t\t\t    Xx ENTER A NEW USER ID xX\n");

		if(nInput != 1)
			printf("\t\t\t\t\t\t Xx INPUT SHOULD BE AN INTEGER xX\n");
	}while(checkUserID(aUserData, *nCountIndex, User.ID) == 1 || nInput != 1 || User.ID < 1); // will loop if the entered user id is not unique

	//password
	do
	{
		fflush(stdin);
		printf("\t\t\t\t\t Password :: ");
		scanf("%s", User.Password);
		fflush(stdin);

		if(strlen(User.Password) > 10)
			printf("\n\t\t\t\t\t\tXx PASSWORD TOO LONG (MAX 10) xX\n\n");
}while(strlen(User.Password) > 10); // will loop if it reached the max password

printf("\t\t\t\t\t Address :: ");
scanf(" ");
fgets(User.Address, MAX30, stdin);
User.Address[strlen(User.Address) - 1 ] = '\0';

//user contact information
do
{
	printf("\t\t\t\t\t Contact Number :: ");
	scanf("%s", User.ContNumber);
	fflush(stdin);
}while(checkContact(User.ContNumber) == 1); // will loop if there is a character

//user name
printf("\t\t\t\t\t Name :: ");
scanf(" ");
fgets(User.Name, MAX20, stdin);
User.Name[strlen(User.Name) - 1 ] = '\0';
fflush(stdin);
printf("\t\t\t\t\t- - - - - - - - - - - - - - - - - - - - - - - - - -\n");
displayUser(User);
printf("\n\t\t\t\t\t=====================================================\n");
cChoice = Confirm();
switch(cChoice)
```

```
		{
			case '1':
						printf("\n\t\t\t\t\t\t\t. . . INFORMATION SAVED . . .\n");
						*nCountIndex = *nCountIndex + 1;
						return User;
						break;
			case '2':
						printf("\n\t\t\t\t\t\t\t    Re-enter user information\n");
						break;
			case '0':
						printf("\n\t\t\t\t\t\t\t    Xx INFORMATION NOT SAVED xX\n");
						break;
		}
	}while (cChoice == '2');
}
/* This function is for displaying the entered information of the user from the Register function
[	@param		(UserInfo)	aUserData = single structure of the user information entered from the Register function  ]
[	@return		no return value
		]	*/
void displayUser(UserInfo aUserData)
{
	int x;
	fflush(stdin);
	printf("\n");
	printf("\t\t\t\t\t");
	for(x = 0; x < 51; x++)
```

```
                printf("=");
        printf("\n\t\t\t\t\t\t\t  ACCOUNT INFORMATION\n");
        printf("\t\t\t\t\t");
        for(x = 0; x < 51; x++)
                printf("=");
        printf("\n");
        printf("\n\t\t\t\t\t\t ID :: %d\n", aUserData.ID);
        printf("\t\t\t\t\t Password :: %s\n", aUserData.Password);
        printf("\t\t\t\t\t Address :: %s\n", aUserData.Address);
        printf("\t\t\t\t\t Contact Number :: %s\n", aUserData.ContNumber);
        printf("\t\t\t\t\t Name :: %s\n", aUserData.Name);
}
/* This function is for sorting the user id in increasing order from the list of users
[       @param          (UserInfo) *aUserData = array list of users                          ]
[                       (int) nCountIndex = index of array list of users           ]
[       @return         (UserInfo) *aUserData = returns the sorted list   of users]          */
void sortID(UserInfo *aUserData, int nCountIndex)
{
   int i, j;
   UserInfo Temp;

   for (i = 0; i < nCountIndex; i++)
   {
      for (j = i + 1; j < nCountIndex; j++)
      {
        if(aUserData[i].ID > aUserData[j].ID)
```

```
                        {
        Temp = aUserData[i];
                                aUserData[i] = aUserData[j];
                                aUserData[j] = Temp;
                            }
                    }
    }
}
/* This function is for user log in
[       @param          (UserInfo) *aUserData = array list of users                                             ]
[                       (int) nCountIndex = index of array list of users                                ]
[                       (int) *uID = entered user id                                                          ]
[       @return         (int) nCheck = returns 1 if found and gets the *uID and -1 if not   ]       */
int Log_In(UserInfo *aUserData, int nCountIndex, int *uID)
{
        string10 cPassword;
        int i, ID, nCheck = -1;
        fflush(stdin);

        printf("\t\t\t\t\t ID :: ");
        scanf("%d", &ID);
        fflush(stdin);

        printf("\t\t\t\t\t Password :: ");
        scanf("%s", cPassword);
        fflush(stdin);
```

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

	for(i = 0; i < nCountIndex && nCheck != 1; i++)
	{
		if(ID == aUserData[i].ID && strcmp(cPassword, aUserData[i].Password) == 0)
		{
			*uID = ID;
			nCheck = 1; // if user is found
		}
	}
	return nCheck;
}
/* This function is for adding a new intem and input the needed informations
[	@param		(ItemInfo) *aItemData = array list of items								]
[			(int) *nSIndex = index of array list of items								]
[	@return		(ItemInfo) Item = returns the entered information to the array of items					]
[			(int) *nSIndex = will increment the index if the user confirmed the item information	]	*/
ItemInfo addItem(ItemInfo *aItemData, int *nSIndex, int sID)
{
	ItemInfo Item;
	char cChoice;
	int nInput;
	do
	{
		fflush(stdin);
		do
```

```
{
        printf("\t\t\t\t\t Product ID :: ");
        nInput = scanf("%d", &Item.iID);
        fflush(stdin);

        if(checkProductID(aItemData, *nSIndex, Item.iID) == 1)
                printf("\n\t\t\t\t\t\t    Xx ENTER A NEW PRODUCT ID xX\n\n");

        if(nInput != 1)
                printf("\t\t\t\t\t\t Xx INPUT SHOULD BE AN INTEGER xX\n");
}while(checkProductID(aItemData, *nSIndex, Item.iID) == 1 || nInput != 1 || Item.iID < 1);

printf("\t\t\t\t\t Item Name :: ");
scanf(" ");
fgets(Item.iName, MAX20, stdin);
Item.iName[strlen(Item.iName) - 1 ] = '\0';

printf("\t\t\t\t\t Category :: ");
scanf(" ");
fgets(Item.iCategory, MAX15, stdin);
Item.iCategory[strlen(Item.iCategory) - 1 ] = '\0';
fflush(stdin);
printf("\t\t\t\t\t Description :: ");
scanf(" ");
fgets(Item.iDescription, MAX30, stdin);
Item.iDescription[strlen(Item.iDescription) - 1 ] = '\0';
```

```
do
{
        printf("\t\t\t\t\t Quantity :: ");
        scanf("%d", &Item.iQty);
        fflush(stdin);
}while(Item.iQty < 0);     // not negative input

do
{
        printf("\t\t\t\t\t Unit Price :: ");
        scanf("%f", &Item.iPrice);
}while(Item.iPrice < 0);  // not negative input

fflush(stdin);
printf("\t\t\t\t\t- - - - - - - - - - - - - - - - - - - - - - - - - -\n");
displayProduct(Item);
printf("\n\t\t\t\t\t====================================================\n");
cChoice = Confirm();
switch(cChoice)
{
        case '1':
                        printf("\n\n\t\t\t\t\t\t\t. . . ITEM SAVED . . .\n");
                        Item.sID = sID;
                        *nSIndex = *nSIndex + 1;
                        return Item;
```

```
				break;
			case '2':
				printf("\n\t\t\t\t\t\t\t      Re-enter item information\n");
				break;
			case '0':
				printf("\n\t\t\t\t\t\t\tXx ITEM NOT SAVED xX\n");
				break;
		}
	}while(cChoice == '2');
}
/* This function is for displaying the entered information of the items from the addItem function
[	@param		(ItemInfo) ItemData = single structure of the item information entered from the addItem function		]
[	@return		no return value ]		*/
void displayProduct(ItemInfo ItemData)
{
	int x;
	fflush(stdin);
	printf("\n");
	printf("\t\t\t\t\t");
	for(x = 0; x < 51; x++)
		printf("=");
	printf("\n\t\t\t\t\t\t\t  ITEM INFORMATION\n");
	printf("\t\t\t\t\t");
	for(x = 0; x < 51; x++)
		printf("=");
	printf("\n");
	printf("\n\t\t\t\t\t ID :: %d\n", ItemData.iID);
	printf("\t\t\t\t\t Name :: %s\n", ItemData.iName);
	printf("\t\t\t\t\t Category :: %s\n", ItemData.iCategory);
	printf("\t\t\t\t\t Description :: %s\n", ItemData.iDescription);
	printf("\t\t\t\t\t Quantity :: %d\n", ItemData.iQty);
	printf("\t\t\t\t\t Price :: %.2f\n", ItemData.iPrice);
}
/* This function is for getting the sellers' items from the list of items
[	@param		(ItemInfo) *aItemData = array list of items						]
[			(int) nSIndex = index of array list of items						]
[			(ItemInfo) *aItem20 = array of the users' bag to be sold / list of items that the user will sell	]
[			(int) ID =  user id of the current user							]
[	@return		(ItemInfo) *aItem20 = returns all the items of the current user				]
[			(int) nCount = returns the index of user list of items to be sold			]	*/
int sellBag20(ItemInfo *aItemData, int nSIndex, ItemInfo *aItem20, int ID)
{
	int i, nCount = 0;
	for(i = 0; i < nSIndex; i++)
	{
		if(ID == aItemData[i].sID)
		{
			aItem20[nCount] = aItemData[i];
			nCount++;
		}
	}
	sortProducts(aItem20, nCount);
	return nCount;

}

/* This function is for showing the list of items of the user either all or one product

[	@param		(ItemInfo) *aItemData = array list of items					]

[			(int) nSIndex = index of array list of items					]

[			(int) nCheck = checks which action to do if -1 it shows all and 1 which will show one product	]

[			(int) pID = product id								]

[	@return		no return value									]
	*/

void showProducts(ItemInfo *aItemData, int nSIndex, int nCheck, int pID)

{

	int x, i, j;


	fflush(stdin);

	printf("\n");

	for(x = 0; x < 160; x++)

		printf("=");

	printf("\n");

	printf("\n%10s %16s %28s %28s %20s %19s %29s\n", "ID", "NAME", "CATEGORY", "DESCRIPTION", "UNIT PRICE", "QUANTITY", "DISCOUNTED PRICE");


	if(nCheck == -1) // showing all the products

	{

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

		{

			printf("%10d\t%-20s \t%-20s \t%-30s %13.2f %16d ", aItemData[i].iID, aItemData[i].iName, aItemData[i].iCategory, aItemData[i].iDescription, aItemData[i].iPrice,
aItemData[i].iQty);
```

```
                if(aItemData[i].iPrice == aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)))
                    printf("\n");
                else
                    printf("%27.2f\n", aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)));
            }
        }
        else // show 1 product
        {
            j = -1;
            do
            {
                j++;
            }while(pID != aItemData[j].iID);
            printf("%10d\t%-20s \t%-20s \t%-30s %13.2f %16d ", aItemData[j].iID, aItemData[j].iName, aItemData[j].iCategory, aItemData[j].iDescription, aItemData[j].iPrice, aItemData[j].iQty);
            if(aItemData[j].iPrice == aItemData[j].iPrice * (1- openDiscount(aItemData[j].iID, aItemData[j].sID)))
                printf("\n");
            else
                printf("%27.2f\n", aItemData[j].iPrice * (1- openDiscount(aItemData[j].iID, aItemData[j].sID)));
        }
        printf("\n");
        for(x = 0; x < 160; x++)
            printf("=");
        printf("\n");
}
/* This function is for sorting the item id in increasing order from the list of items
[       @param      (ItemInfo) *aItemData = array list of items                        ]
[                                   (int) nSIndex = index array list of items                                        ]
[           @return           (ItemInfo) *aItemData = returns the sorted list of items  ]            */
void sortProducts(ItemInfo *aItemData, int nSIndex)
{
   int i, j;
   ItemInfo Temp;
   for (i = 0; i < nSIndex; i++)
   {
      for (j = i + 1; j < nSIndex; j++)
      {
           if(aItemData[i].iID > aItemData[j].iID)
                         {
           Temp = aItemData[i];
                                   aItemData[i] = aItemData[j];
                                   aItemData[j] = Temp;
                                }
                      }
   }
}
/* This function is for adding and removing the quantity of the specific item that the user wishes to change
[           @param            (ItemInfo) *aItem20 = array of the users' bag to be sold  ]
[                             (int) nIndex20 = index of array users' bag to be sold                 ]
[                             (ItemInfo) *aItemData = array list of items                                        ]
[                             (int) nSIndex = index of array list of items                                ]
[                             (int) pID = entered product id                                                        ]
[           @return           (ItemInfo) *aItem20      & *aItemData = returns the replenished quantity of the item to the structure .iQty          ]            */
```

```
void Replenish_Reduce(ItemInfo *aItem20, int nIndex20, ItemInfo *aItemData, int nSIndex, int pID, int nCheck)
{
        int i, j, nQuantity;

        if(nCheck == 1)
        {
                for(i = 0; i < nIndex20; i++)       // this loop is for the seller's bag
                {
                        if(pID == aItem20[i].iID) // this is for checking the entered product id to the bag's item id
                        {
                                do
                                {
                                        printf("\t\t\t\t\t Add Quantity :: ");
                                        scanf("%d", &nQuantity);
                                }while(nQuantity < 0);
                                aItem20[i].iQty += nQuantity;
                        }
                }
                for(j = 0; j < nSIndex; j++)        // this loop is for the array of items
                if(pID == aItemData[j].iID)         // this is for checking the entered product id to the array of items
                        aItemData[j].iQty += nQuantity;
        }
        else
        {
                for(i = 0; i < nIndex20; i++)       // this loop is for the seller's bag
                {
                        if(pID == aItem20[i].iID) // this is for checking the entered product id to the bag's item id
                        {
                        do
                                {
                                        printf("\t\t\t\t\t Reduce Quantity :: ");
                                        scanf("%d", &nQuantity);
                                }while(nQuantity < 0 || nQuantity > aItem20[i].iQty);
                                aItem20[i].iQty -= nQuantity;
                        }
                }
                for(j = 0; j < nSIndex; j++)        // this loop is for the array of items
                if(pID == aItemData[j].iID)         // this is for checking the entered product id to the array of items
                        aItemData[j].iQty -= nQuantity;
        }
        printf("\n");
}
/* This function is for changing the price of the specific item that the user wishes to change
[       @param          (ItemInfo) *aItem20 = array of the users' bag to be sold  ]
[                       (int) nIndex20 = index of array users' bag to be sold                   ]
[                       (ItemInfo) *aItemData = array list of items                                             ]
[                       (int) nSIndex = index of array list of items                                    ]
[                       (int) pID = entered product id                                                          ]
[       @return         (ItemInfo) *aItem20     & *aItemData = returns the newly entered price of the item to the structure .iPrice     ]       */
void changePrice(ItemInfo *aItem20, int nIndex20, ItemInfo *aItemData, int nSIndex, int pID)
{
        int i, j;
```

```
	float fPrice;


	for(i = 0; i < nIndex20; i++)		// this loop is for the seller's bag
	{
		if(pID == aItem20[i].iID) // this is for checking the entered product id to the bag's item id
		{
			do
			{
				printf("\t\t\t\t\t Enter New Price :: ");
				scanf("%f", &fPrice);
			}while(fPrice < 0);
			aItem20[i].iPrice = fPrice;
		}
	}
	for(j = 0; j < nSIndex; j++)		// this loop is for the array of items
		if(pID == aItemData[j].iID)		// this is for checking the entered product id to the array of items
			aItemData[j].iPrice = fPrice;
	printf("\n");
}
/* This function is for changing the name of the specific item that the user wishes to change
[	@param		(ItemInfo) *aItem20 = array of the users' bag to be sold  ]
[			(int) nIndex20 = index of array users' bag to be sold			]
[			(ItemInfo) *aItemData = array list of items						]
[			(int) nSIndex = index of array list of items						]
[			(int) pID = entered product id										]
[	@return		(ItemInfo) *aItem20	& *aItemData = returns the newly entered name of the item to the structure .iName	]		*/
```

```
void changeName(ItemInfo *aItem20, int nIndex20, ItemInfo *aItemData, int nSIndex, int pID)
{
	int i,j;
	for(i = 0; i < nIndex20; i++)		// this loop is for the seller's bag
	{
		if(pID == aItem20[i].iID) // this is for checking the entered product id to the bag's item id
		{
			printf("\t\t\t\t\t Enter New Item Name :: ");
			scanf(" ");
			fgets(aItem20[i].iName, MAX20, stdin);
			aItem20[i].iName[strlen(aItem20[i].iName) - 1] = '\0';

			for(j = 0; j < nSIndex; j++)		// this loop is for the array of items
				if(pID == aItemData[j].iID)		// this is for checking the entered product id to the array of items
					strcpy(aItemData[j].iName, aItem20[i].iName);
		}
	}
	printf("\n");
}
/* This function is for changing the category of the specific item that the user wishes to change
[	@param		(ItemInfo) *aItem20 = array of the users' bag to be sold  ]
[			(int) nIndex20 = index of array users' bag to be sold			]
[			(ItemInfo) *aItemData = array list of items						]
[			(int) nSIndex = index of array list of items					]
[			(int) pID = entered product id									]
[	@return		(ItemInfo) *aItem20	& *aItemData = returns the newly entered category of the item to the structure .iCategory]	*/
```

```
void changeCategory(ItemInfo *aItem20, int nIndex20, ItemInfo *aItemData, int nSIndex, int pID)
{
        int i,j;
        for(i = 0; i < nIndex20; i++)       // this loop is for the seller's bag
        {
                if(pID == aItem20[i].iID) // this is for checking the entered product id to the bag's item id
                {
                        printf("\t\t\t\t\t Enter New Product Category :: ");
                        scanf(" ");
                        fgets(aItem20[i].iCategory, MAX15, stdin);
                        aItem20[i].iCategory[strlen(aItem20[i].iCategory) - 1] = '\0';

                        for(j = 0; j < nSIndex; j++)        // this loop is for the array of items
                                if(pID == aItemData[j].iID)       // this is for checking the entered product id to the array of items
                                        strcpy(aItemData[j].iCategory, aItem20[i].iCategory);
                }
        }
        printf("\n");
}
/* This function is for changing the description of the specific item that the user wishes to change
[       @param          (ItemInfo) *aItem20 = array of the users' bag to be sold  ]
[                       (int) nIndex20 = index of array users' bag to be sold           ]
[                       (ItemInfo) *aItemData = array list of items                                      ]
[                       (int) nSIndex = index of array list of items                                ]
[                       (int) pID = entered product id                                                         ]
[       @return         (ItemInfo) *aItem20     & *aItemData = returns the newly entered description of the item to the structure .iDescription  ]          */
```

```
void changeDescription(ItemInfo *aItem20, int nIndex20, ItemInfo *aItemData, int nSIndex, int pID)
{
	int i,j;
	for(i = 0; i < nIndex20; i++)		//this loop is for the seller's bag
	{
		if(pID == aItem20[i].iID) // this is for checking the entered product id to the bag's item id
		{
			printf("\t\t\t\t\t Enter Product Description :: ");
			scanf(" ");
			fgets(aItem20[i].iDescription, MAX30, stdin);
			aItem20[i].iDescription[strlen(aItem20[i].iDescription) - 1] = '\0';

			for(j = 0; j < nSIndex; j++)		//this loop is for the array of items
				if(pID == aItemData[j].iID)		// this is for checking the entered product id to the array of items
					strcpy(aItemData[j].iDescription, aItemData[i].iDescription);
		}
	}
	printf("\n");
}
/* This function is for showing the low product whose quantity falls below 5
[	@param		(ItemInfo) *aItemData = array list of items			]
[			(int) nSIndex = index of array list of items		]
[	@return		no return value								]		*/
void showLowProducts(ItemInfo *aItemData, int nSIndex)
{
	char cOpt = 'N';
	int i, x;
	fflush(stdin);
	for(x = 0; x < 160; x++)
		printf("=");
	printf("\n");
	x = 0;
	for(i = 0; i < nSIndex && cOpt != 'X' && cOpt != 'x'; i++)
	{
		if(aItemData[i].iQty < 5) // if the item's quantity is below 5
		{
			printf("\npress [N] to see the next\npress [X] to exit the view\n");
			printf("\n%10s %16s %28s %28s %20s %19s %29s\n", "ID", "NAME", "CATEGORY", "DESCRIPTION", "UNIT PRICE", "QUANTITY", "DISCOUNTED PRICE");
			printf("%10d\t%-20s \t%-20s \t%-30s %13.2f %16d ", aItemData[i].iID, aItemData[i].iName, aItemData[i].iCategory, aItemData[i].iDescription, aItemData[i].iPrice,
aItemData[i].iQty);
			if(aItemData[i].iPrice == aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)))
				printf("\n");
			else
				printf("%27.2f\n", aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)));

			x++;
			do
			{
				printf("\n Enter choice: ");
				scanf(" %c", &cOpt);
			}while(cOpt != 'N' && cOpt != 'n'&& cOpt != 'X' && cOpt != 'x');
		}
	}
	if(x == 0)
		printf("\n\t\t\t\t\t\t    Xx NO PRODUCTS BELOW 5 QUANTITY xX\n");
	printf("\n");
}
/* This function is for adding a discount in all or specific item it also saves the discount to a binary file <product id>.dsc
[	@param		(ItemInfo) *aItemData = array list of items		]
[			(int) nSIndex = index of array list of items		]
[			(int) ID = ID of the user					]
[	@return		no return value							]	*/
void addDiscount(ItemInfo *aItemData, int nSIndex, int ID)
{
	int i, pID = 0;
	float fDiscount;
	char cChoice;
	FILE *pFile;
	string15 cID;
	DiscountInfo Discount;

	do
	{
		printf("\n\t\t\t\t\t\t Enter Discount in (%%) :: ");
		scanf("%f", &fDiscount);
	}while(fDiscount < 0);

	do
```

```
{
	printf("\n\t\t\t\t\t\t\t\t<<>><<>><<<>><<>><<>><<>>\n");
	printf("\t\t\t\t\t\t\t<<                    >>\n");
	printf("\t\t\t\t\t\t\t<< Apply to All Items? >>\n");
	printf("\t\t\t\t\t\t\t<<                    >>\n");
	printf("\t\t\t\t\t\t\t<< [1] Yes            >>\n");
	printf("\t\t\t\t\t\t\t<< [2] No             >>\n");
	printf("\t\t\t\t\t\t\t<<                    >>\n");
	printf("\t\t\t\t\t\t\t<<>><<>><<<>><<>><<>><<>>\n");
	printf("\n\t\t\t\t\t\t\t\tChoice :: ");
	scanf(" %c", &cChoice);

	if(cChoice != '1' && cChoice != '2')
		printf("\n\t\t\t\t\t\t\t  Xx INVALID CHOICE xX\n");

}while(cChoice != '1' && cChoice != '2');

if(cChoice == '1')		// if yes
	pID = ID;		// copies the user to pID

else	// if certain item
	pID = enterPID(aItemData, nSIndex);	// copies the entered id to pID

for(i = 0; i < nSIndex; i++)
{
	Discount.iID = 0;
```

```
		if(pID == aItemData[i].iID)		// check item ID
		{
			Discount.Discount = fDiscount / 100;
			Discount.iID = aItemData[i].iID;
			Discount.sID = aItemData[i].sID;
		}
		else if(pID == aItemData[i].sID)	// check item seller ID
		{
			Discount.Discount = fDiscount / 100;
			Discount.iID = aItemData[i].iID;
			Discount.sID = aItemData[i].sID;
		}
		if(Discount.iID > 0)
		{
			itoa(Discount.iID, cID, 10);		// itoa converts (int) to string
			pFile = fopen (strcat(cID,".dsc"), "wb");
			fwrite(&Discount, sizeof(struct DiscountTag), 1, pFile);
			fclose(pFile);
		}
	}
	printf("\n\n\t\t\t\t");
	for(i = 0; i < 43; i++)
		printf("- ");
	printf("\n");
}
/* This function is for removing a discount in all or specific item
```

```
[       @param      (ItemInfo) *aItemData = array list of items          ]
[                   (int) nSIndex = index of array list of items       ]
[                   (int) ID = ID of the user                                    ]
[       @return     no return value                                                      ]       */
void removeDiscount(ItemInfo *aItemData, int nSIndex, int ID)
{
        int i, pID = 0;
        char cChoice;
        string15 cID;

        do
        {
                printf("\n\t\t\t\t\t\t\t<<>><<>><<<>><<>><<>><<>>\n");
                printf("\t\t\t\t\t\t\t<<                 >>\n");
                printf("\t\t\t\t\t\t\t<< Apply to All Items? >>\n");
                printf("\t\t\t\t\t\t\t<<                 >>\n");
                printf("\t\t\t\t\t\t\t<< [1] Yes           >>\n");
                printf("\t\t\t\t\t\t\t<< [2] No            >>\n");
                printf("\t\t\t\t\t\t\t<<                 >>\n");
                printf("\t\t\t\t\t\t\t<<>><<>><<<>><<>><<>><<>>\n");
                printf("\n\t\t\t\t\t\t\tChoice :: ");
                scanf(" %c", &cChoice);

                if(cChoice != '1' && cChoice != '2')
                        printf("\n\t\t\t\t\t\t\t  Xx INVALID CHOICE xX\n");
        }while(cChoice != '1' && cChoice != '2');


        if(cChoice == '1')
        {
                pID = ID;
                for(i = 0; i < nSIndex; i++)
                {
                        if(pID == aItemData[i].sID)
                        {
                                itoa(aItemData[i].iID, cID, 10);
                                if (remove(strcat(cID, ".dsc")) == 0)       // removes the .dsc file
                                    printf("\n\t\t\t\t\t\t Xx DISCOUNT REMOVED SUCCESSFULLY FOR ITEM < %d > xX", aItemData[i].iID);
                        }
                }
        }
        else
        {
                pID = enterPID(aItemData, nSIndex);
                if(checkProductID(aItemData, nSIndex, pID) == 1)
                {
                        itoa(pID, cID, 10);
                        if (remove(strcat(cID, ".dsc")) == 0)
                            printf("\n\t\t\t\t\t\t Xx DISCOUNT REMOVED SUCCESSFULLY FOR ITEM < %d > xX", pID);
                }
        }
        printf("\n\n\t\t\t\t");
```

```
	for(i = 0; i < 43; i++)
		printf("- ");
	printf("\n");
}
/* This function is for opening the discount of the item in binary file
[	@param		(ItemInfo) *aItemData = array list of items					]
[			(int) nSIndex = index of array list of items				]
[			(int) ID = ID of the user							]
[	@return		(float) fDiscount = returns the discount of the price	]		*/
float openDiscount(int pID, int sID)
{
	float fDiscount = 0;
	FILE *pFile;
	string15 ID;
	DiscountInfo Discount;

	itoa(pID, ID, 10);		// itoa converts (int) to string
	pFile = fopen (strcat(ID,".dsc"), "rb");

	if(pFile)
	{
		fread(&Discount, sizeof(struct DiscountTag), 1, pFile);
		fDiscount = Discount.Discount;
	}
	fclose(pFile);
	return fDiscount;
```

```
}
/* This function is for entering the product id
[       @param      (ItemInfo) = array list of items                    ]
[                   (int) nSIndex = index of array list of items        ]
[       @return     (int) ID = if the user is found and -1 if not       ]       */
int enterPID(ItemInfo *aItemData, int nSIndex)
{
        int ID;

        printf("\n\n\t\t\t\t Enter Product ID :: ");
        scanf("%d", &ID);

        if(checkProductID(aItemData, nSIndex, ID) == 1)
                return ID;
        else
                printf("\n\t\t\t\t\t\t    Xx PRODUCT ID NOT FOUND xX\n");
        return -1;
}
/* This function is for displaying the products of the users one at a time
[       @param      (ItemInfo) *aItemData = array list of items                ]
[                   (int) nSIndex = index of array list of items        ]
[                   (int) uID = user id        of the current user                    ]
[                   (UserInfo) *aUserData = array list of users                ]
[                   (int) nCountIndex = index of array list of users    ]
[       @return     no return value ]       */
void allProducts(ItemInfo *aItemData, int nSIndex, int uID, UserInfo *aUserData, int nCountIndex)
```

```
{
	char cOpt = 'N';
	int i, j, x, nCheck = -1;
	fflush(stdin);

	for(j = 0; j < nCountIndex; j++)
	{
		nCheck = 0;
		for(i = 0; i < nSIndex; i++)
			if(aUserData[j].ID == aItemData[i].sID)
				nCheck = 1;

		if(nCheck && aUserData[j].ID != uID)
		{
			for(x = 0; x < 160; x++)
				printf("=");
			printf("\n press [N] to see the next\n press [X] to exit the view\n\n\n");
			printf("Seller : %d, %s\n\n", aUserData[j].ID, aUserData[j].Name);
			printf("\n%10s %16s %28s %28s %20s %19s %29s\n", "ID", "NAME", "CATEGORY", "DESCRIPTION", "UNIT PRICE", "QUANTITY", "DISCOUNTED PRICE");
			nCheck = -1;
			for(i = 0; i < nSIndex; i++)
				if(aUserData[j].ID == aItemData[i].sID)
				{
					printf("%10d\t%-20s \t%-20s \t%-30s %13.2f %16d ", aItemData[i].iID, aItemData[i].iName, aItemData[i].iCategory, aItemData[i].iDescription,
aItemData[i].iPrice, aItemData[i].iQty);
					if(aItemData[i].iPrice == aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)))
```

```
                        printf("\n");
                    else
                        printf("%27.2f\n", aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)));
                }
            do
            {
                printf("\n Enter choice: ");
                scanf(" %c", &cOpt);
                if(cOpt == 'X' || cOpt == 'x')
                    j = nCountIndex;
            }while(cOpt != 'N' && cOpt != 'n' && cOpt != 'X' && cOpt != 'x');
        }
    }
    printf("\n");
}
/* This function is for displaying the products of the users with discounts
[   @param      (ItemInfo) *aItemData = array list of items             ]
[               (int) nSIndex = index of array list of items        ]
[               (int) uID = user id     of the current user             ]
[               (UserInfo) *aUserData = array list of users             ]
[               (int) nCountIndex = index of array list of users    ]
[   @return     no return value ]   */
void viewDiscount(ItemInfo *aItemData, int nSIndex, int uID, UserInfo *aUserData, int nCountIndex)
{
    char cOpt = 'N';
    int i, j, x, nCheck = -1, nCount = 0;
```

```
	fflush(stdin);
	for(j = 0; j < nCountIndex; j++)
	{
		nCheck = 0;
		for(i = 0; i < nSIndex; i++)
			if(aItemData[i].iPrice != aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)))
				if(aUserData[j].ID == aItemData[i].sID)
					nCheck = 1;

		if(nCheck && aUserData[j].ID != uID)	//	if found
		{
			for(x = 0; x < 160; x++)
				printf("=");
			printf("\n press [N] to see the next\n press [X] to exit the view\n\n\n");
			printf("Seller : %d, %s\n\n", aUserData[j].ID, aUserData[j].Name);
			printf("\n%10s %16s %28s %28s %20s %19s %29s\n", "ID", "NAME", "CATEGORY", "DESCRIPTION", "UNIT PRICE", "QUANTITY", "DISCOUNTED PRICE");
			nCheck = -1;
			for(i = 0; i < nSIndex; i++)
				if(aItemData[i].iPrice != aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)))
					if(aUserData[j].ID == aItemData[i].sID)
					{
						printf("%10d\t%-20s \t%-20s \t%-20s %23.2f %16d %27.2f\n", aItemData[i].iID, aItemData[i].iName, aItemData[i].iCategory,
aItemData[i].iDescription, aItemData[i].iPrice, aItemData[i].iQty, aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)));
						nCount++;
					}
			do
```

```
				{
					printf("\n Enter choice: ");
					scanf(" %c", &cOpt);
					if(cOpt == 'X' || cOpt == 'x')
						j = nCountIndex;
				}while(cOpt != 'N' && cOpt != 'n' && cOpt != 'X' && cOpt != 'x');
			}
	}
	if(nCount < 1)
		printf("\n\t\t\t\t\t\t     Xx NO DISCOUNTED PRODUCTS xX\n");
	printf("\n");
}
/* This function is for showing the items of the entered specific seller the user wishes to see
[	@param		(ItemInfo) *aItemData = array list of items			]
[			(int) nSIndex = index of array list of items		]
[			(int) uID = user id of the current user				]
[	@return		no return value									]		*/
void specificSeller(ItemInfo *aItemData, int nSIndex, int uID)
{
	int i, x, enterID, nCheck = -1;

	printf("\n\t\t\t\t Search Seller ID :: ");
	scanf("%d", &enterID);

	for(i = 0; i < nSIndex; i++)
		if(enterID == aItemData[i].sID)
```

```
                        nCheck = 1;


        if(nCheck != 1)
                printf("\n\t\t\t\t\t\tXx SELLER ID NOT FOUND xX\n");


        else
        {
                if(enterID != uID)
                {
                        printf("\n%10s %16s %28s %28s %20s %19s %29s\n", "ID", "NAME", "CATEGORY", "DESCRIPTION", "UNIT PRICE", "QUANTITY", "DISCOUNTED PRICE");
                        for(i = 0; i < nSIndex ; i++)
                        {
                                if(aItemData[i].sID == enterID)
                                {
                                        printf("%10d\t%-20s \t%-20s \t%-30s %13.2f %16d ", aItemData[i].iID, aItemData[i].iName, aItemData[i].iCategory, aItemData[i].iDescription,
aItemData[i].iPrice, aItemData[i].iQty);
                                        if(aItemData[i].iPrice == aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)))
                                                printf("\n");
                                        else
                                                printf("%27.2f\n", aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)));
                                }
                        }
                        printf("\n");
                        for(x = 0; x < 160; x++)
                                printf("=");
                        printf("\n");
                }
                else
                        printf("\n\t\t\t\t\t\tXx GO TO SELL MENU TO CHECK YOUR ITEMS xX\n");


        }
        printf("\n");
}
/* This function is for showing the items of the entered category the user wishes to see
[       @param          (ItemInfo) *aItemData = array list of items                     ]
[                       (int) nSIndex = index of array list of items            ]
[                       (int) uID = user id of the current user                         ]
[       @return         no return value                                                                 ]       */
void searchCategory(ItemInfo *aItemData, int nSIndex, int uID)
{
        char cOpt = 'N';
        int i, x, nCheck = -1;
        string20 enterCat, TempCat;     // tempCat stores the category of the items in lowercase
        fflush(stdin);

        printf("\n\t\t\t Search Product Category :: ");
        scanf(" ");
        fgets(enterCat, 15, stdin);
        enterCat[strlen(enterCat) - 1] = '\0';

        for(i = 0; i < nSIndex && cOpt != 'X' && cOpt != 'x'; i++)
        {
```

```
                    strcpy(TempCat, aItemData[i].iCategory);
                    if(strcmp(strlwr(enterCat), strlwr(TempCat)) == 0)                          // compares the entered category from the list of items
                    {
                            if(aItemData[i].sID != uID)                         // will do the display the following if item seller id and current user is not the same
                            {
                                    printf("\n");
                                    for(x = 0; x < 160; x++)
                                            printf("=");
                                    printf("\npress [N] to see the next\npress [X] to exit the view\n");
                                    printf("\n%10s %16s %28s %28s %20s %19s %29s\n", "ID", "NAME", "CATEGORY", "DESCRIPTION", "UNIT PRICE", "QUANTITY", "DISCOUNTED PRICE");
                                    printf("%10d\t%-20s \t%-20s \t%-30s %13.2f %16d ", aItemData[i].iID, aItemData[i].iName, aItemData[i].iCategory, aItemData[i].iDescription, aItemData[i].iPrice,
aItemData[i].iQty);

                                    if(aItemData[i].iPrice == aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)))
                                            printf("\n");
                                    else
                                            printf("%27.2f\n", aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)));

                                    nCheck = 1;
                                    do
                                    {
                                            printf("\nEnter choice: ");
                                            scanf(" %c", &cOpt);
                                            fflush(stdin);
                                    }while(cOpt != 'N' && cOpt != 'n' && cOpt != 'X' && cOpt != 'x' );
                            }
```

```
                }
        }
        if(nCheck != 1)
        printf("\n\t\t\t\t\t\tXx PRODUCT CATEGORY NOT FOUND xX\n");
        printf("\n");
}
/* This function is for showing the items of the entered name the user wishes to see
[       @param          (ItemInfo) *aItemData = array list of items             ]
[                       (int) nSIndex = index of array list of items        ]
[                       (int) uID = user id of the current user                 ]
[       @return         no return value                                                 ]       */
void searchName(ItemInfo *aItemData, int nSIndex, int uID)
{
        char cOpt = 'N';
        int i, x, nCheck = -1;
        string20 enterName, TempName; // TempName stores the name of the products in lowercase
        fflush(stdin);
        printf("\n\t\t\t Search Item Name :: ");
        scanf(" ");
        fgets(enterName, 15, stdin);
        enterName[strlen(enterName) - 1] = '\0';

        for(i = 0; i < nSIndex && cOpt != 'X' && cOpt != 'x'; i++)
        {
                strcpy(TempName, aItemData[i].iName);
                if(strstr(strlwr(TempName), strlwr(enterName)) != NULL)         // strstr is for searching the substring and strlwr makes both the item name and temporary name lowercase
```

```
			{
				if(aItemData[i].sID != uID)
				{
					printf("\n");
					for(x = 0; x < 160; x++)
						printf("=");
					printf("\npress [N] to see the next\npress [X] to exit the view\n\n\n");
					printf("\n%10s %16s %28s %28s %20s %19s %29s\n", "ID", "NAME", "CATEGORY", "DESCRIPTION", "UNIT PRICE", "QUANTITY", "DISCOUNTED PRICE");
					printf("%10d\t%-20s \t%-20s \t%-30s %13.2f %16d ", aItemData[i].iID, aItemData[i].iName, aItemData[i].iCategory, aItemData[i].iDescription, aItemData[i].iPrice, 
aItemData[i].iQty);
					if(aItemData[i].iPrice == aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)))
						printf("\n");
					else
						printf("%27.2f\n", aItemData[i].iPrice * (1- openDiscount(aItemData[i].iID, aItemData[i].sID)));


					nCheck = 1;
					do
					{
						printf("\nEnter choice: ");
						scanf(" %c", &cOpt);
						fflush(stdin);
					}while(cOpt != 'N' && cOpt != 'n' && cOpt != 'X' && cOpt != 'x');
				}


			}
		}
```

```
	if(nCheck != 1)
		printf("\n\t\t\t\t\t\tXx ITEM NAME NOT FOUND xX\n");
	printf("\n");
}
/* This function is for displaying the cart of the user
[	@param		(CartInfo) *aCartData  = array list of items from the users' cart    ]
[			(int) nCartIndex = index of the array of items from the users' cart		]
[	@return		no return value														]		*/
void displayCart(CartInfo *aCartData, int nCartIndex)
{
	int i;
	printf("\n  < < YOUR CART > >\n");

	printf("%15s %15s %15s %18s %20s %20s %15s %20s\n\n", "SELLER ID", "PRODUCT ID", "NAME", "QUANTITY", "UNIT PRICE", "TOTAL PRICE", "LESS", "SUBTOTAL");
	for(i = 0; i < nCartIndex ; i++)
	{
		printf("%14d %15d\t  %-20s %10d %20.2f %20.2f ", aCartData[i].cItem.sID, aCartData[i].cItem.iID, aCartData[i].cItem.iName, aCartData[i].cQty, aCartData[i].cItem.iPrice,
aCartData[i].cItem.iPrice * aCartData[i].cQty);
		if(aCartData[i].cItem.iPrice * aCartData[i].cQty - aCartData[i].cPrice == 0 )
			printf("%36.2f\n", aCartData[i].cItem.iPrice * aCartData[i].cQty);

		else
			printf("%18.2f %17.2f\n", aCartData[i].cItem.iPrice * aCartData[i].cQty - aCartData[i].cPrice , aCartData[i].cPrice);
	}
	printf("\n");
	for(i = 0; i < 77; i++)
```

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

/* This function is for adding a certain product and enters the quantity that the user wanted to add to the cart it should exist from the list of items
[	@param		(CartInfo) *aCartData = array list of items from the users' cart				]
[			(int) *nCartIndex = index of the array list of items from the users' cart			]
[			(ItemInfo) *aItemData = array list of items							]
[			(int) nSIndex = index of array list of items							]
[			(int) bID = id of the current user								]
[	@return		(CartInfo) Cart = returns the entered information to the array of users' cart		]
[			(int) *nCartIndex = will increment the index if the user confiremed the item information he wanted to add to the cart	]	*/
CartInfo addCart(CartInfo *aCartData, int *nCartIndex, ItemInfo *aItemData, int nSIndex, int bID)
{
	int i, k, enterID, enterQty, getIndex = 0;
	int nCheck = *nCartIndex;
	CartInfo Cart;
	char cChoice;

	do
	{
		printf("\n\t\t\t\t\t Enter Product ID :: ");
		scanf("%d", &enterID);

		if(checkProductID(aItemData, nSIndex, enterID) != 1)
			printf("\n\t\t\t\t\t    Xx PRODUCT ID NOT FOUND xX\n");
```

```
else if(nCheck + 1 > 10 && checkCProductID(aCartData, *nCartIndex, enterID) != 1)
        printf("\n\t\t\t\t\t\t    Xx CART IS FULL xX\n");


else
{
        for(i = 0; i < nSIndex; i++)
        {
                if(aItemData[i].iID == enterID && aItemData[i].iQty > 0)  // to check if the entered id exists
                {
                        if(aItemData[i].sID != bID)          // makes sure that the user cant add his own product to the cart
                        {
                                for(k = 0; k < *nCartIndex; k++)   // if product already in cart, gets the item index from the cart
                                        if(aCartData[k].cItem.iID == enterID)
                                                getIndex = k;


                                do
                                {
                                        printf("\t\t\t\t\t Enter Product Quantity :: ");
                                        scanf("%d", &enterQty);
                                }while(enterQty < 0);


                                if(enterQty <= aItemData[i].iQty || aCartData[getIndex].cItem.iQty + enterQty <= aItemData[i].iQty )
                                {
                                        if(enterQty == 0)
                                                printf("\n\t\t\t\t\tXx QUANTITY SET TO 0. THIS ITEM WONT BE CHECKED OUT xX\n");
        Cart.cItem = aItemData[i];
        Cart.cQty = enterQty;
        Cart.cPrice = aItemData[i].iPrice * Cart.cQty * (1- openDiscount(aItemData[i].iID,aItemData[i].sID));

        printf("\n PRODUCT INFORMATION :\n");
        showProducts(aItemData, nSIndex, 1, enterID);
        printf("\n -- Quantity :: %d", Cart.cQty);
        printf("\n -- Subtotal :: %.2f\n", Cart.cPrice);
        cChoice = Confirm();
        switch(cChoice)
        {
                case '1':
                                printf("\n\t\t\t\t\t       . . . PRODUCT ADDED TO CART . . .\n");
                                if(checkCProductID(aCartData, *nCartIndex, enterID) == 1)          // if product already exists in the cart
                                {
                                        aCartData[getIndex].cQty += Cart.cQty;
                                        aCartData[getIndex].cItem = aItemData[i];
                                        aCartData[getIndex].cPrice = aItemData[i].iPrice * aCartData[getIndex].cQty * (1-
openDiscount(aItemData[i].iID,aItemData[i].sID));
                                }
                                else
                                {
                                        *nCartIndex = *nCartIndex + 1;
                                        return Cart;
                                }
```

```
                                                            break;
                                                case '2':
                                                            printf("\n\t\t\t\t\t\t    RE-ENTER PRODUCT TO ADD CART\n");
                                                            break;
                                                case '0':
                                                            printf("\n\t\t\t\t\t\t\t\t Xx PRODUCT NOT ADDED xX\n");
                                                            break;
                                          }
                                    }
                                    else
                                          printf("\n\t\t\t\t\t\t\t\tXx INVALID QUANTITY xX\n");
                              }
                              else
                                    printf("\n\t\t\t\t\t\t\t    Xx PRODUCT ID NOT FOUND xX\n");
                        }
                        else if(aItemData[i].iID == enterID && aItemData[i].iQty == 0)
                                    printf("\n\t\t\t\t\t\t\t\tXx ITEM SOLD OUT xX\n");
                  }
            }
      }while(cChoice == '2');
      return Cart;
}
/* This function is for comparing the list of items and the users' cart if changes were made after the user added the item to the cart
[     @param      (CartInfo) *aCartData = array list of items from the users' cart                  ]
[                 (int) nCartIndex = index of the array list of items from the users' cart     ]
[                 (ItemInfo) *aItemData = array list of items                                                                   ]
```

```
[				(int) nSIndex = index of array list of items								]
[	@return		no return value													]	*/
void compareItem(CartInfo *aCartData, int nCartIndex, ItemInfo *aItemData, int nSIndex)
{
	int i, j, nCount = 0;
	float newPrice = 0;
	for(i = 0; i < nCartIndex; i++)
	{
		for(j = 0; j < nSIndex; j++)
		{
			if(aCartData[i].cItem.iID == aItemData[j].iID)	// to check if the id of items in the users' cart is the same as the one in the list of all items of all sellers
			{
				if(strcmp(aCartData[i].cItem.iCategory, aItemData[j].iCategory) != 0)	// to check if the the items' category in the cart and the list of all items of all sellers is the 
same
				{
					printf("\n\t\t\t\t\tCATEGORY OF PRODUCT ID < %d > HAS BEEN CHANGED\n", aCartData[i].cItem.iID);
					printf("\t\t\t\t\t-> PREVIOUS CATEGORY : %s\n", aCartData[i].cItem.iCategory);
					printf("\t\t\t\t\t-> NEW CATEGORY : %s\n", aItemData[j].iCategory);
					nCount++;
				}
				if(strcmp(aCartData[i].cItem.iDescription, aItemData[j].iDescription) != 0)		// to check if the the items' description in the cart and the list of all items of all sellers 
is the same
				{
					printf("\n\t\t\t\t\tDESCRIPTION OF PRODUCT ID < %d > HAS BEEN CHANGED\n", aCartData[i].cItem.iID);
					printf("\t\t\t\t\t-> PREVIOUS DESCRIPTION : %s\n", aCartData[i].cItem.iDescription);
					printf("\t\t\t\t\t-> NEW DESCRIPTION : %s\n", aItemData[j].iDescription);
```

```
                        nCount++;
                    }
                    if(strcmp(aCartData[i].cItem.iName, aItemData[j].iName) != 0)    // to check if the the items' name in the cart and the list of all items of all sellers is the same
                    {
                        printf("\n\t\t\t\t\tNAME OF PRODUCT ID < %d > HAS BEEN CHANGED\n", aCartData[i].cItem.iID);
                        printf("\t\t\t\t\t-> PREVIOUS NAME : %s\n", aCartData[i].cItem.iName);
                        printf("\t\t\t\t\t-> NEW NAME : %s\n", aItemData[j].iName);
                        nCount++;
                    }
                    if(aCartData[i].cItem.iPrice != aItemData[j].iPrice)
                    {
                        printf("\n\t\t\t\t\tPRICE OF PRODUCT ID < %d > HAS BEEN CHANGED\n", aCartData[i].cItem.iID);// to check if the the items' price in the cart and the list of all
items of all sellers is the same
                        printf("\t\t\t\t\t-> PREVIOUS PRICE : %.2f\n", aCartData[i].cItem.iPrice);
                        printf("\t\t\t\t\t-> NEW PRICE : %.2f\n", aItemData[j].iPrice);
                        nCount++;
                    }
                    if(aCartData[i].cQty > aItemData[j].iQty) // to check if the the items' quantity in the cart and the list of all items of all sellers is the same
                    {
                        printf("\n\t\t\t\t\tQUANTITY OF PRODUCT ID < %d > IS INSUFFICIENT\n", aCartData[i].cItem.iID);
                        printf("\n\t\t\t\t\tINSUFFICIENT PRODUCT WILL NOT BE CHECKED OUT\n");
                        printf("\t\t\t\t\t-> CART QUANTITY : %d\n", aCartData[i].cItem.iQty);
                        printf("\t\t\t\t\t-> AVAILABLE QUANTITY : %d\n", aItemData[j].iQty);
                        nCount++;
                    }
                    newPrice = aItemData[j].iPrice * aCartData[i].cQty * (1 - openDiscount(aItemData[j].iID, aItemData[j].sID));
```

```
                                        if(roundUp(aCartData[i].cPrice) != roundUp(newPrice))   // to check if the the items' subtotal in the cart and the list of items of all sellers is the same
                                        {
                                                printf("\n\t\t\t\t\tSUBTOTAL PRICE IN CART FOR PRODUCT ID < %d > HAS CHANGED\n", aCartData[i].cItem.iID);
                                                printf("\t\t\t\t\t-> PREVIOUS SUBTOTAL : %.2f\n", aCartData[i].cPrice);
                                                printf("\t\t\t\t\t-> NEW SUBTOTAL : %.2f\n", newPrice);
                                                nCount++;
                                        }
                                }
                        }
                }
                if(nCount > 0)
                        printf("\n\t\t\t\t\t<< UPDATED PRODUCT DATA WILL BE APPLIED ON CHECKOUT\n\t\t\t\t\t   GO TO EDIT CART MENU TO EDIT THE ITEM >>");
                printf("\n");
}
/* This function is for rounding up the price of the float
[               @param          (float) fNum = price                            ]
[               @return         (float) fRound = round up                       ]               */
float roundUp(float fNum)
{
   float fRound;
                fRound = (int)(fNum * 100 + .5);
   return (float)fRound / 100;
}
/* This function is for removing the item of the entered seller from the users' cart
[               @param          (CartInfo) *aCartData = array list of items from the users' cart                        ]
[                               (int) *nCartIndex = index of the array list of items from the users' cart       ]
```

```
[	@return		(CartInfo) *aCartData = updates the list of items of the user's cart		]
[			(int) *nCartIndex = will decrement after removing the seller				]	*/
void removeSeller(CartInfo *aCartData, int *nCartIndex)
{
	int i, entersID;
	CartInfo aTemp;

	printf("\n\t\t\t\t\t Enter Seller ID :: ");
	scanf("%d", &entersID);

	if(checkCSellerID(aCartData, *nCartIndex, entersID) == 1)
	{
		for(i = 0; i < *nCartIndex; i++)
		{
			if(aCartData[i].cItem.sID == entersID)
			{
				aTemp = aCartData[i];
				aCartData[i] = aCartData[*nCartIndex - 1];
				aCartData[*nCartIndex - 1] = aTemp;
				*nCartIndex= *nCartIndex - 1;
				i--;
			}
		}
	}
	else
		printf("\n\t\t\t\t\t\tXx SORRY, SELLER ID NOT FOUND xX\n");
	printf("\n");
}
/* This function is for removing the item of the entered id from the users' cart
[	@param		(CartInfo) *aCartData = array list of items from the users' cart							]
[			(int) *nCartIndex = index of the array list of items from the users' cart						]
[			(int) pID = product if; if -1 the user will enter the product id else it will skip the enter product id		]
[	@return		(CartInfo) *aCartData = updates the list of items of the user's cart							]
[			(int) *nCartIndex = will decrement after removing the item								]	*/
void removeItem(CartInfo *aCartData, int *nCartIndex, int pID)
{
	int i;
	CartInfo aTemp;
	int nID = pID;
	if(nID == -1)
	{
		printf("\n\t\t\t\t\t Enter Product ID :: ");
		scanf("%d", &nID);
	}

	if(checkCProductID(aCartData, *nCartIndex, nID) == 1)
	{
		for(i = 0; i < *nCartIndex; i++)
		{
			if(aCartData[i].cItem.iID == nID)
			{
				aTemp = aCartData[i];
```

```
                    aCartData[i] = aCartData[*nCartIndex - 1];
                    aCartData[*nCartIndex - 1] = aTemp;
                    *nCartIndex= *nCartIndex - 1;
                    i--;
                }
            }
        }
        else
            printf("\n\t\t\t\t\t\tXx SORRY, PRODUCT ID NOT FOUND xX\n");
}
/* This function is for editing the quantity of the item in the users' cart
[       @param      (CartInfo) *aCartData = array list of items from the users' cart                ]
[                   (int) *nCartIndex = index of the array list of items from the users' cart   ]
[                   (ItemInfo) *aItemData = array list of items                                          ]
[                   (int) nSIndex = index of array list of items                                       ]
[       @return     (CartInfo) *aCartData = updates the list of items of the users' cart if the user wishes to remove the product which has a quantity of 0 ]
                    (int) *nCartIndex = will decrement after the user confirmed to remove the item with 0 quantity
        ]       */
void editQty(CartInfo *aCartData, int *nCartIndex, ItemInfo *aItemData, int nSIndex)
{
        int i, j, enterpID, enterQty;
        char cChoice;
        fflush(stdin);
        printf("\n\t\t\t\t\t Enter Product ID :: ");
        scanf("%d", &enterpID);
```

```
if(checkCProductID(aCartData, *nCartIndex, enterpID) != 1)
        printf("\n\t\t\t\t\tXx PRODUCT ID NOT FOUND IN CART xX\n");


for(i = 0; i < nSIndex; i++)
{
        if(aItemData[i].iID == enterpID)
        {
                for(j = 0; j < *nCartIndex; j++)
                {
                        if(aCartData[j].cItem.iID == enterpID)
                        {
                                do
                                {
                                        printf("\t\t\t\t\t Enter New Quantity :: ");
                                        scanf("%d", &enterQty);
                                }while(enterQty < 0);

                                if(aItemData[i].iQty >= enterQty)
                                {
                                        aCartData[j].cQty = enterQty;
                                        aCartData[j].cPrice = aItemData[i].iPrice * enterQty * (1- openDiscount(aItemData[i].iID,aItemData[i].sID));
                                }
                                else
                                        printf("\n\t\t\t\t\t\t    Xx QUANTITY NOT AVAILABLE xX\n");

                                if(enterQty == 0)
```

```
{
        do
        {
                fflush(stdin);
                printf("\n\t\t\t\t\t    <<>><<>><<<>><<>><<>><<>><<>><<>><<>><<>>\n");
                printf("\t\t\t\t\t    <<                              >>\n");
                printf("\t\t\t\t\t    << Do You Want to Remove This Item?    >>\n");
                printf("\t\t\t\t\t    <<                              >>\n");
                printf("\t\t\t\t\t    << [1] Yes                          >>\n");
                printf("\t\t\t\t\t    << [2] No                           >>\n");
                printf("\t\t\t\t\t    <<                              >>\n");
                printf("\t\t\t\t\t    <<>><<>><<<>><<>><<>><<>><<>><<>><<>><<>>\n");
                printf("\n\t\t\t\t\t     Choice :: ");
                scanf(" %c", &cChoice);
                printf("\n");

                if(cChoice != '1' && cChoice != '2')
                        printf("\n\t\t\t\t\t\t\t  Xx INVALID CHOICE xX\n");

        }while(cChoice != '1' && cChoice != '2');

        if(cChoice == '1')
        {
                printf("\n\t\t\t\t\t\tXx ITEM REMOVED FROM THE CART xX\n");
                removeItem(aCartData, nCartIndex, enterpID);
        }
```

```
						else
							printf("\n\t\t\t\t\tXx QUANTITY SET TO 0. THIS ITEM WONT BE CHECKED OUT xX\n");
					}
				}
			}
		}
	}
	printf("\n");
}
/* This function is for asking the user to enter the month, day, year
[	@param		(int) *nMonth, *nDay, *nYear = pointer month, day, year						]
[	@return		(int) *nMonth, *nDay,  *nYear = returns the entered month, day, year	]		*/
void getDate(int *nMonth, int *nDay, int *nYear)
{
	int nTempMonth, nTempDay, nTempYear, Month;
	do
	{
		fflush(stdin);
		printf("\n\t\t\t\t\t\t( INPUT SHOULD BE IN NUMERIC !! )\n");
		printf("\t\t\t\t\t Enter Month : ");
		scanf("%d", &nTempMonth);
		fflush(stdin);
		printf("\t\t\t\t\t Enter Day : ");
		scanf("%d", &nTempDay);
		fflush(stdin);
		printf("\t\t\t\t\t Enter Year : ");
```

```
        scanf("%d", &nTempYear);
        fflush(stdin);
        switch(nTempMonth)
        {
            case 1: case 3: case 5: case 7: case 8:   case 10: case 12:       Month = 31; break;
            case 4: case 6: case 9: case 11:  Month = 30; break;
            case 2:
                    if(nTempYear % 4 == 0)
                    {
                        if(nTempYear % 100 != 0)
                            Month = 29;
                        else
                        {
                            if(nTempYear % 400 == 0)
                                Month = 29;
                            else
                                Month = 28;
                        }
                    }
                    else
                        Month = 28;
                    break;
        }
    }while(nTempMonth < 1 || nTempMonth > 12 || nTempDay < 1 || nTempDay > Month || nTempYear < 1);

    *nMonth = nTempMonth;
```

```
	*nDay = nTempDay;
	*nYear = nTempYear;
}
/* This function is for asking the use whether to confirm, re-enter or cancel the entered date
[	@param		(TransacInfo) TransacData = a single structure of the transaction information			]
[	@return		(TransacInfo) TransacData = returns the date information needed for the transaction	]		*/
TransacInfo confirmDate(TransacInfo TransacData)
{
	char cChoice;
	DateInfo nD;

	do
	{
		getDate(&nD.nMonth, &nD.nDay, &nD.nYear);
		cChoice = Confirm();
		switch(cChoice)
		{
			case '1':
						printf("\n\t\t\t\t\t\t  . . . SUCCESSFULLY ENTERED DATE . . .\n");
						TransacData.tD = nD;
						return TransacData;
						break;
			case '2':
						printf("\n\t\t\t\t\t\t    RE-ENTER DATE\n");
						break;
			case '0':
```

```
                                                printf("\n\t\t\t\t\t\t    Xx DATE NOT ENTERED xX\n");
                                                TransacData.tD.nMonth = -1;
                                                return TransacData;
                                                break;
                    }
        }while(cChoice == '2');
}
/* This function is for sorting the date in increasing order from the list of transaction
[       @param          (TransacInfo) *TransacData = array list of transaction                          ]
[                       (int) nTransacIndex = index array list of transaction                           ]
[       @return         (TransacInfo) *TransacData = returns the sorted list of transactions    ]       */
void sortDate(TransacInfo *aTransacData, int nTransacIndex)
{
        TransacInfo temp;
        int i, j;

        for(i = 0; i < nTransacIndex; i++)
                for(j = 0; j < nTransacIndex; j++)
                        if(aTransacData[i].tD.nDay + aTransacData[i].tD.nMonth * 100 + aTransacData[i].tD.nYear * 10000 < aTransacData[j].tD.nDay + aTransacData[j].tD.nMonth * 100 +
aTransacData[j].tD.nYear * 10000)
                        {
                                temp = aTransacData[i];
                                aTransacData[i] = aTransacData[j];
                                aTransacData[j] = temp;
                        }
}
```

```
/* This function is for getting the information from the entered specific seller
[       @param      (CartInfo) *aCartTemp = array list of items from the users' cart                              ]
[                   (int) nCartTemp = index of array list of items from the users' cart                              ]
[                   (ItemInfo) *aItemData = array list of items                                                        ]
[                   (int) nSIndex = index of array list of items                                                     ]
[                   (TransacInfo) Transac = a single structure of the transaction information                ]
[                   (Int) sID = seller id; if -1 the user will enter the seller id else it will skip the enter seller id  ]
[       @return     (TransacInfo) tempTransac = returns the transaction information          ]     */
TransacInfo transacSeller(CartInfo *aCartTemp, int nCartTemp, ItemInfo *aItemData, int nSIndex, TransacInfo Transac, int sID)
{
        int i, j, nCount = 0;
        float Subtotal = 0;
        CartInfo aCartData[10];
        int nCartIndex = 0;
        TransacInfo tempTransac;
        fflush(stdin);

        tempTransac.tD = Transac.tD;

        if(sID == -1)
        {
                printf("\t\t\t\t\t Enter Seller ID :: ");
                scanf("%d", &sID);
        }

        for(i = 0; i < nCartTemp; i++)      // checks the cart array
```

```
{
        for(j = 0; j < nSIndex; j++)          // checks the item array
        {
                if(aItemData[j].iID == aCartTemp[i].cItem.iID)     // if found
                {
                        if(aCartTemp[i].cQty <= aItemData[j].iQty && aCartTemp[i].cQty > 0)
                        {
                                aCartData[nCartIndex] = aCartTemp[i];   // copies the information to the temporary array aCartData
                                nCartIndex++;
                        }
                }
        }
}

if(checkCSellerID(aCartData, nCartIndex, sID) == 1) // check seller id
{
        nCount = 0;
        for(i = 0; i < nCartIndex && nCount < 5; i++)        // makes sure that the receipt can only accomodate 5 items
        {
                if(sID == aCartData[i].cItem.sID) // if entered seller id and seller id in the cart is the same
                {
                        for(j = 0; j < nSIndex; j++)
                        {
                                if(aItemData[j].iID == aCartData[i].cItem.iID) // if the items in the item list and cart list is the same
                                {
                                        if(aCartData[i].cQty <= aItemData[j].iQty) // if the product in cart is less than or equal to the item list
                        {
                            tempTransac.iId[nCount] = aCartData[i].cItem.iID;
                            strcpy(tempTransac.iName[nCount], aItemData[j].iName);
                            tempTransac.iPrice[nCount] = aItemData[j].iPrice;
                            tempTransac.tQty[nCount] = aCartData[i].cQty;
                            tempTransac.sId = aItemData[j].sID;
                            tempTransac.iDiscount[nCount] = openDiscount(aCartData[i].cItem.iID, aCartData[i].cItem.sID);
                            aItemData[j].iQty = aItemData[j].iQty - aCartData[i].cQty;
                            Subtotal += aCartData[i].cQty * aItemData[j].iPrice * (1 - tempTransac.iDiscount[nCount]);
                            nCount++;
                            j = nSIndex;    // to stop the for loop of j
                        }
                    }
                }
            }
        }
        for(i = 0; i < nCount; i++)
            removeItem(aCartTemp, &nCartTemp, tempTransac.iId[i]);
        tempTransac.tPrice = Subtotal;
    }
    tempTransac.tIndex = nCount;
    return tempTransac;
}
/* This function is for transacting a specific item
[   @param      (CartInfo) *aCartData = array list of items from the users' cart
[               (int) nCartIndex = index of the array list of items from the users' cart
[                   (ItemInfo) *aItemData = array list of items
[                   (int) nSIndex = index of array list of items
[                   (TransacInfo) TransacData = a single structure of the transaction information (int) nID = product id; if -1 the user will enter the seller id else it will skip the enter product id
[       @return     (TransacInfo) tempTransac = returns the transaction information ]       */
TransacInfo transacItem(CartInfo *aCartData, int nCartIndex, ItemInfo *aItemData, int nSIndex, TransacInfo TransacData)
{
        int i, j;
        int nCID, nCount = 0;
        float Subtotal = 0;

        fflush(stdin);
        printf("\t\t\t\t Enter Product ID :: ");
        scanf("%d", &nCID);

        if(checkCProductID(aCartData, nCartIndex, nCID) == 1) // check product id
        {
                for( i = 0; i < nCartIndex && nCount < 5; i++)
                {
                        if(nCID == aCartData[i].cItem.iID) // if entered product id and item id is the same
                        {
                                for(j = 0; j < nSIndex; j++)
                                {
                                        if(nCID == aItemData[j].iID)
                                        {
                                                if(aCartData[i].cQty <= aItemData[j].iQty && aCartData[i].cQty > 0)
                                                {
```

```
                                        TransacData.iId[nCount] = aCartData[i].cItem.iID;
                                        strcpy(TransacData.iName[nCount], aItemData[j].iName);
                                        TransacData.iPrice[nCount] = aItemData[j].iPrice;
                                        TransacData.tQty[nCount] = aCartData[i].cQty;
                                        TransacData.sId = aItemData[j].sID;
                                        TransacData.iDiscount[nCount] = openDiscount(aCartData[i].cItem.iID, aCartData[i].cItem.sID);
                                        aItemData[j].iQty = aItemData[j].iQty - aCartData[i].cQty;
                                        Subtotal += aCartData[i].cQty * aItemData[j].iPrice * (1 - TransacData.iDiscount[nCount]);
                                        nCount++;
                                        j = nSIndex;       // to stop the loop of j
                                    }
                                    else
                                        printf("\n\t\t\t\t\t\t\t Xx ITEM CANT BE CHECKED OUT\n\t\t\t\t\t\t EDIT YOUR PRODUCT QUANTITY IN YOUR CART xX\n");
                                }
                            }
                        }
                }
                for( i = 0; i < nCount; i++)
                        removeItem(aCartData, &nCartIndex, TransacData.iId[i]);
                TransacData.tPrice = Subtotal;
        }
        else
                printf("\n\t\t\t\t\t\t\t   Xx PRODUCT ID NOT FOUND xX\n\n");
        TransacData.tIndex = nCount;
        return TransacData;
}
```

```
/* This function is for copying the needed information which is the seller and buyer information to the transaction information
[           @param          (UserInfo) *UserData = array list of users                                                        ]
[                           (int) nCountIndex = index of array list of users                                          ]
[                           (TransacInfo) Transac = a single structure of the transaction information ]
[                           (int) nID = user id         of the current user                                                       ]
[           @return         (TransacInfo) Transac = returns the needed information for the transaction        ]          */
TransacInfo completeInfo(UserInfo *UserData, int nCountIndex, TransacInfo Transac, int nID)
{
            int i;

            for(i = 0; i < nCountIndex; i++)
            {
                        if(Transac.sId == UserData[i].ID)
                        {
                                    strcpy(Transac.sName, UserData[i].Name);
                                    strcpy(Transac.sAddress, UserData[i].Address);
                        }
                        if(nID == UserData[i].ID)
                        {
                                    Transac.bId = nID;
                                    strcpy(Transac.bName, UserData[i].Name);
                                    strcpy(Transac.bAddress, UserData[i].Address);
                        }
            }
            return Transac;
}
```

```
/* This function is for displaying the receipt
[       @param      (TransacInfo) Transac = a single structure of the transaction information ]
[       @return     no return value                                                          ]      */
void displayReceipt(TransacInfo Transac)
{
        int i, x;
        printf("\n\t\t+");
        for(x = 0; x < 125; x++)
                printf("-");
        printf("+");

        printf("\n\n\t\t\t\t\t\t\t\tH A V E N B R O O K  I N C .\n");
        printf("\t\t\t\t\t\t\t\t    %s, %d\n", Transac.sName, Transac.sId);
        printf("\t\t\t\t\t\t\t\t    %s\n\n\n\n", Transac.sAddress);
        printf("\t\t\t %s          \t\t\t\t\t\t  %s : %d / %d / %d", "BUYER INFO", "TRANSACTION DATE", Transac.tD.nMonth,Transac.tD.nDay,Transac.tD.nYear);
        printf("\n\t\t\t ID NO :  %d", Transac.bId);
        printf("\n\t\t\t NAME  :  %s", Transac.bName);
        printf("\n\t\t\t ADDRESS : %s", Transac.bAddress);
        printf("\n\n\n\t  %20s %18s %25s %20s %20s %20s\n\n\t\t   ", "PRODUCT ID", "ITEM NAME", "QUANTITY", "UNIT PRICE","DISCOUNT (%)", "SUBTOTAL");
        for(x = 0; x < 121; x++)
                printf("=");
        printf("\n\n");
        for(i = 0; i < Transac.tIndex; i++)
        {
                printf("\t\t   %10d\t   %-20s %18d %20.2f ", Transac.iId[i], Transac.iName[i], Transac.tQty[i], Transac.iPrice[i]);
```

```
			if(Transac.iDiscount[i] * 100 != 0.0)
				printf("%19.2f%% %20.2f\n", Transac.iDiscount[i] * 100, (1 - Transac.iDiscount[i]) * Transac.tQty[i] * Transac.iPrice[i]);
			else
				printf("%41.2f\n", (1 - Transac.iDiscount[i]) * Transac.tQty[i] * Transac.iPrice[i]);
		}
		printf("\n\t\t   ");
		for(x = 0; x < 121; x++)
			printf("=");
		printf("\n\n\t\t\t\t\t\t\t\t   %55s : %12.2f", "TOTAL", Transac.tPrice);
		printf("\n\n");

		printf("\n\t\t+");
		for(x = 0; x < 125; x++)
			printf("-");
		printf("+");
		printf("\n\n");
}
/* This function is for displaying all the users
[	@param		(UserInfo) *aUserData = array list of users				]
[			(int) nCountIndex = indaex of array list of users   ]
[	@return		no return value										]	*/
void adminUsers(UserInfo *aUserData, int nCountIndex)
{
	int i;
	fflush(stdin);
	printf("\n\t%10s %14s %18s %30s %37s\n\n ", "ID", "PASSWORD", "NAME", "ADDRESS", "CONTACT NUMBER");
	for(i = 0; i < nCountIndex; i++)
		printf("\t%10d \t%-10s \t%-22s \t%-22s \t%36s\n", aUserData[i].ID, aUserData[i].Password, aUserData[i].Name, aUserData[i].Address, aUserData[i].ContNumber);
	printf("\n");
	for(i = 0; i < 80; i++)
		printf("- ");
	printf("\n\n");
}
/* This function is for displaying all the sellers
[	@param		(ItemInfo) *aItemData = array list of items				]
[			(int) nSIndex = index of array list of items			]
[			(UserInfo) *aUserData = array list of users				]
[			(int) nCountIndex = index of array list of users	]
[	@return		no return value								]		*/
void adminSellers(ItemInfo *aItemData, int nSIndex, UserInfo *aUserData, int nCountIndex)
{
	int i, j, nBagIndex = 0;
	ItemInfo aItem20[20];	// putting the items to be sold of the seller
	fflush(stdin);
	printf("\n\t%10s %14s %18s %30s %37s %30s\n\n ", "ID", "PASSWORD", "NAME", "ADDRESS", "CONTACT NUMBER", "ITEM FOR SALE");
	for(i = 0; i < nCountIndex; i++)	// loop for the users
	{
		nBagIndex = 0;
		for(j = 0; j < nSIndex; j++)		// loop for the items
		{
			if(aItemData[j].sID == aUserData[i].ID)
			{
```

```
                                        nBagIndex = sellBag20(aItemData, nSIndex, aItem20, aItemData[j].sID);

                                        printf("\t%10d \t%-10s \t%-22s \t%-22s \t%36s %25d\n", aUserData[i].ID, aUserData[i].Password, aUserData[i].Name, aUserData[i].Address,
aUserData[i].ContNumber, nBagIndex);

                                        j = nSIndex;

                                }

                        }

                }

                printf("\n");

                for(i = 0; i < 80; i++)

                        printf("- ");

                printf("\n\n");

}

/* This function is for displaying the total amount of all tranasctions from the entered date

[       @param          no input parameter      ]

[       @return         no return value ]       */

void adminTotalSales()

{

        TransacInfo tempTransac[100];  // temporary array for structure of transaction

        int i, nTransacIndex = 0;

        float fPrice = 0;


        openTransac(tempTransac, &nTransacIndex);


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

                fPrice += tempTransac[i].tPrice;
```

```
	if(nTransacIndex > 0)
		printf("\n\t\t\t\t\t    The total amount of all transactions : %.2f", fPrice);


	printf("\n\n\t\t\t");
	for(i = 0; i < 45; i++)
		printf("- ");
	printf("\n\n");
}
/* This function is for getting the index of the seller
[	@param		(UserInfo) *aUserData = array list of users					]
[			(TransacInfo) *TransacData = array of transaced items			]
[			(int) nTransacIndex = index of the array of transaced items		]
[	@return		(int) nIndex = returns the index of the seller				]	*/
int TransacSID(UserInfo *aUserData, TransacInfo *TransacData, int nTransacIndex)
{
	int i,j, nCheck = 0, nIndex = 0;


	for(i = 0; i < nTransacIndex; i++)
	{
		nCheck = 0;
		for(j = 0; j < nIndex; j++)		// checks if the the seller id is already in the temporary seller list
		{
			if(aUserData[j].ID == TransacData[i].sId)
				nCheck = 1;			// if yes, nCheck = 1
		}
```

```
                if(nCheck != 1)   // if nCheck = 0 add seller to the temporary seller list
                {
                        aUserData[nIndex].ID = TransacData[i].sId;
                        strcpy(aUserData[nIndex].Name, TransacData[i].sName);
                        nIndex++;
                }
        }
        return nIndex;
}
/* This function is for displaying the total sales for each seller
[       @param          (int) nID = id of the user                              ]
[                       (string20) UserName = name of the user]
[       @return         no return value                                         ]       */
void adminSellerSales(int nID, string20 UserName)
{
        TransacInfo tempTransac[100];  // used for getting the structure of transactions
        UserInfo aTempUser[100];        // temporary array for structure of user
        int nTransacIndex = 0, nCountIndex = 0;
        int i, j, k, x, nCheck = 0;
        float fPrice = 0;

        openTransac(tempTransac, &nTransacIndex);

        if(nID == -1)       // gets all the seller sales of all sellers
                nCountIndex = TransacSID(aTempUser, tempTransac, nTransacIndex);
        else
```

```
{
	nCountIndex = 1;
	aTempUser[0].ID = nID;
	strcpy(aTempUser[0].Name, UserName);

	for(i = 0; i < nTransacIndex; i++)
		if(tempTransac[i].sId == nID)
			nCheck++;
	if(nCheck == 0)
	{
		nTransacIndex = 0;
		printf("\n\n\t\t\t\t\t\t   Xx NO TRANSACTIONs TO SHOW xX\n\n");
	}
}

if(0 < nTransacIndex)
{
	printf("\n\t\t\t");
	for(x = 0; x < 100; x++)
		printf("=");
	printf("\n");
	printf("\t\t\t\t %20s %20s\t\t%20s\n\n", "SELLER ID", "SELLER NAME", "AMOUNT");
	for(i = 0; i < nCountIndex; i++)     // gets all the seller sales of all sellers
	{
		fPrice = 0;
		for(j = 0; j < nTransacIndex; j++)
```

```
                    if(aTempUser[i].ID == tempTransac[j].sId)
                        fPrice += tempTransac[j].tPrice;

                if(fPrice != 0)
                    printf("\t\t\t\t %19d\t   %-20s %30.2f\n", aTempUser[i].ID, aTempUser[i].Name, fPrice);
            }

            if(nID != -1)       // gets the sales of a certain seller
            {
                sortDate(tempTransac, nTransacIndex);
                printf("\n\n\t\t\tSUMMARY OF PRODUCTs SOLD\n");
                printf("\n\t\t\t%9s %27s %25s %18s %15s\n", "DATE", "BUYER NAME", "ITEM NAME", "QUANTITY", "PRICE");
                for(i = 0; i < nTransacIndex; i++)
                {
                    if(nID == tempTransac[i].sId)
                    {
                        for(k = 0; k < tempTransac[i].tIndex; k++)
                            printf("\t\t\t%2d / %2d / %4d\t\t%-20s\t%-20s %12d %18.2f\n", tempTransac[i].tD.nMonth, tempTransac[i].tD.nDay, tempTransac[i].tD.nYear,
tempTransac[i].bName, tempTransac[i].iName[k], tempTransac[i].tQty[k], tempTransac[i].tQty[k] * tempTransac[i].iPrice[k] * (1 - tempTransac[i].iDiscount[k]));
                    }
                }
            }
        }
        printf("\n\t\t");
        for(x = 0; x < 56; x++)
            printf("- ");
```

```
	printf("\n\n");
}
/* This function is for displaying the total amount for each buyer in table format and total amount bought in the duration
[	@param		(UserInfo) *aUserData = array list of users				]
[			(int) nCountIndex = index of array list of users	]
[	@return		no return value								]	*/
void adminShopaholics(UserInfo *aUserData, int nCountIndex)
{
	TransacInfo tempTransac[100];
	int i,j, x, nTransacIndex = 0;
	float fPrice = 0;

	openTransac(tempTransac, &nTransacIndex);

	if(nTransacIndex > 0)
	{
		printf("\n\t\t\t");
		for(x = 0; x < 100; x++)
			printf("=");
		printf("\n");

		printf("\t\t\t\t %20s %20s\t\t%20s\n\n", "BUYER ID", "BUYER NAME", "AMOUNT");

		for(i = 0; i < nCountIndex; i++)
		{
			fPrice = 0;
```

```
                                for(j = 0; j < nTransacIndex; j++)
                                {
                                        if(aUserData[i].ID == tempTransac[j].bId)
                                                fPrice += tempTransac[j].tPrice;
                                }
                                if(fPrice != 0)
                                        printf("\t\t\t\t %19d\t   %-20s %30.2f\n", aUserData[i].ID, aUserData[i].Name, fPrice);
                        }
                }
                printf("\n\t\t");
                for(x = 0; x < 59; x++)
                        printf("- ");
                printf("\n\n");
}
/* This function is for displaying all the transaced items of the user in table form
[       @param          (int) ID = id of the current user   ]
[       @return         no return value ]           */
void userReceipt(int ID)
{
        int i, j, nTransacIndex = 0, nCheck = 0;
        float fPrice = 0;
        TransacInfo Transac[100];

        openTransac(Transac, &nTransacIndex);

        for(i = 0; i < nTransacIndex; i++)
```

```
            if(ID == Transac[i].bId)

                nCheck = 1;

        if(nCheck == 0)

            printf("\n\n\t\t\t\t\t\t\tXx NO PRODUCTS TO SHOW xX\n\n");

        else

        {

            sortDate(Transac, nTransacIndex);

            printf("\n\n\t\t\tSUMMARY OF PRODUCTs BOUGHT\n");

            printf("\n\t\t\t%9s %27s %25s %18s %15s\n", "DATE", "SELLER NAME", "ITEM NAME", "QUANTITY", "PRICE");

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

            {

                if(ID == Transac[i].bId)

                {

                    for(j = 0; j < Transac[i].tIndex; j++)

                        printf("\t\t\t%2d / %2d / %4d\t\t%-20s\t%-20s %12d %18.2f\n", Transac[i].tD.nMonth, Transac[i].tD.nDay, Transac[i].tD.nYear, Transac[i].sName,
Transac[i].iName[j], Transac[i].tQty[j], Transac[i].tQty[j] * Transac[i].iPrice[j] * (1 - Transac[i].iDiscount[j]));


                    fPrice += Transac[i].tPrice;

                }

            }

            printf("\n\t\t\tTOTAL : %.2f\n", fPrice);

        }

        printf("\n\t\t");

        for(i = 0; i < 56; i++)

            printf("- ");

        printf("\n\n");
```

```
}


int main()
{
	system("COLOR 0B");
	UserInfo aUserList[100]; // array for users
	UserInfo aUser; // single structure data of user
	int nUIndex = 0; // count index for user list

	ItemInfo aSItems[100 * 20]; // array for items
	int nUSIndex = 0; // count index for item list

	ItemInfo aSBag20[20]; // bag of the certain user with max 20 items
	int nBagIndex = 0;

	CartInfo aCartList[10]; // cart of user ; load from binary file
	int nCartIndex = 0;

	TransacInfo TransacList; //fix index //own or all

	char cMMChoice;		// main menu option
	char cUAction;	// user action option
	char cBChoice; // buy option
	char cECChoice; // edit cart option
	char cSChoice; // sell option
	char cESChoice; // edit stock option
char cCheckOut; // check out products option

char cSTChoice; // show transac menu option

char cAChoice;  // admin option


int nUID;           // user id

int sID;  // product id


string10 cAdmin; // admin password


int i, j, x, nCheck = 0;


openUsers(aUserList, &nUIndex); //check existing users

openItems(aSItems, &nUSIndex); // check exisiting items


do

{

        sortID(aUserList, nUIndex);

        sortProducts(aSItems, nUSIndex);

        cMMChoice = MainMenu();

        switch (cMMChoice)

        {


                case '1':       //system ("cls");

                                if(nUIndex < 100) // max number of users

                                {

                                        printf("\n\t\t\t\t\t- - - - - - - - - R E G I S T E R - - - - - - - - -\n");
                aUserList[nUIndex] = Register(aUserList, &nUIndex);

                sortID(aUserList, nUIndex);

            }

            else

                printf("\n\t\t\t\t\t\t    Xx MAXIMUM USERS REACHED xX\n");


            break;

case '2'://system ("cls");

            printf("\n\t\t\t\t\t- - - - - - - - - - L O G   I N - - - - - - - - - -\n");

            if(Log_In(aUserList, nUIndex, &nUID) != 1)          // if user not found

                printf("\n\t\t\t\t\t\t\tXx LOG IN ERROR xX\n");

            else // if user found

            {

                for(x = 0; x < nUIndex; x++)                    // loop for getting the user info

                    if(aUserList[x].ID == nUID)

                        aUser = aUserList[x];

                nCartIndex = 0;

                openCart(aCartList, &nCartIndex, nUID);// open the users' cart

                do

                {

                    nBagIndex = sellBag20(aSItems, nUSIndex, aSBag20, nUID);        // gets the items to be sold by the user

                    printf("\n\n\t\t\t\t\t+ - - - - - - -  U S E R   M E N U  - - - - - - - +\n");

                    cUAction = UserMenu();

                    switch(cUAction)

                    {
```

```
                                        fflush(stdin);
                                        case '1':
                                                    do
                                                    {
                                                        printf("\n\t\t\t\t\t+ - - - - - - - - ACTION: S E L L - - - - - - - - +\n");

                                                        cSChoice = SellMenu();
                                                        switch(cSChoice)
                                                        {
                                                            fflush(stdin);
                                                            case '1':
                                                                        if(nBagIndex < 20)          // max number to be sold by seller
                                                                        {
                                                                            printf("\n\t\t\t\t\t\t- - - - - - - - - A D D   I T E M S - - - - -
- - -\n");

                                                                            aSItems[nUSIndex] = addItem(aSItems, &nUSIndex,
nUID);

                                                                            sortProducts(aSItems, nUSIndex);
                                                                            nBagIndex = sellBag20(aSItems, nUSIndex, aSBag20,
nUID);

                                                                        }
                                                                        else
                                                                            printf("\n\t\t\t\t\t\t\t\tXx BAG IS FULL xX\n");

                                                                        break;
                                                            case '2':
                                                                        fflush(stdin);
```

```
if(nBagIndex == 0)
        printf("\n\t\t\t\t\t\t\tXx NO ITEMS TO SHOW xX\n");
else
{
        printf("\n < < YOUR ITEMS > >");
        showProducts(aSBag20, nBagIndex, -1, -1);       //
shows all your items

        sID = enterPID(aSBag20, nBagIndex);     // returns
the product id that the user wish to edit

        if(sID != -1)       // if seller id is found
        {

                do
                {
                        showProducts(aSBag20, nBagIndex,
1, sID);  // shows the product you wish to edit

                        cESChoice = editStock();
                        switch(cESChoice)
                        {

                                case '1':

        printf("\n\t\t\t\t\t\t~ ~ ~ ~ ~ ~ ~ ~  R E P L E N I S H  ~ ~ ~ ~ ~ ~ ~ ~\n\n");

        Replenish_Reduce(aSBag20, nBagIndex, aSItems, nUSIndex, sID, 1);

        break;

                                case '2':
```

```
printf("\n\t\t\t\t\t~ ~ ~ ~ ~ ~ ~ ~ ~ ~ R E D U C E ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~\n\n");

Replenish_Reduce(aSBag20, nBagIndex, aSItems, nUSIndex, sID, -1);

break;
                                                                        case '3':

printf("\n\t\t\t\t\t~ ~ ~ ~ ~ ~ ~ ~  C H A N G E  P R I C E  ~ ~ ~ ~ ~ ~\n\n");

changePrice(aSBag20, nBagIndex, aSItems, nUSIndex, sID);

break;
                                                                        case '4':


printf("\n\t\t\t\t\t~ ~ ~ ~ ~  C H A N G E   I T E M   N A M E  ~ ~ ~ ~ ~\n\n");

changeName(aSBag20, nBagIndex, aSItems, nUSIndex, sID);

break;
                                                                        case '5':

printf("\n\t\t\t\t\t~ ~ ~ ~ ~ ~  C H A N G E   C A T E G O R Y  ~ ~ ~ ~ ~\n\n");

changeCategory(aSBag20, nBagIndex, aSItems, nUSIndex, sID);

break;
                                                                        case '6':
```

```
printf("\n\t\t\t\t\t~ ~ ~ ~  C H A N G E  D E S C R I P T I O N  ~ ~ ~ ~\n\n");

changeDescription(aSBag20, nBagIndex, aSItems, nUSIndex, sID);

break;
                                                                                        case '0':

fflush(stdin);

printf("\n\t\t\t\t\t\t    . . . FINISH EDITING . . .\n");

break;
                                                                                        default:

printf("\n\t\t\t\t\t\t\t Xx INVALID INPUT xX\n");

                                                                                    }
                                                                                }while(cESChoice != '0');
                                                                            }
                                                                        }

                                                                        break;
                                                            case '3':
                                                                        fflush(stdin);
                                                                        if(nBagIndex == 0) // if empty bag
                                                                            printf("\n\t\t\t\t\t\t\tXx NO ITEMS TO SHOW xX\n");
                                                                        else
                                                                        {
```

```
                printf("\n");
                for(x = 0; x < 160; x++)
                        printf("=");
                printf("\n\t\t\t\t= = = = = = = = = = = = = = = = P R O D
U C T  L I S T = = = = = = = = = = = = = = = =");

                showProducts(aSBag20, nBagIndex, -1, -1);
        }
        break;
case '4':
        fflush(stdin);
        if(nBagIndex == 0)
                printf("\n\t\t\t\t\t\t\tXx NO ITEMS TO SHOW xX\n");
        else
        {
                printf("\n");
                for(x = 0; x < 160; x++)
                        printf("=");

                printf("\n\t\t\t\t= = = = = = = = = = = = = = = = MY
LOW STOCK PRODUCT  = = = = = = = = = = = = = = = =\n");

                showLowProducts(aSBag20, nBagIndex);
        }
        break;
case '5':
        fflush(stdin);
        if(nBagIndex == 0) // if empty bag
```

```
                        printf("\n\t\t\t\t\t\t\tXx NO ITEMS TO ADD
DISCOUNT xX\n");
                else
                {
                        showProducts(aSBag20, nBagIndex, -1, -1);       //
shows all your items
                        printf("\n\t\t\t\t+ - - - - - - - - - - - - - - - -  DISCOUNT:
ADD  - - - - - - - - - - - - - - - - +\n");
                        addDiscount(aSBag20, nBagIndex, nUID);
                }

                break;
        case '6':
                fflush(stdin);
                if(nBagIndex == 0) // if empty bag
                        printf("\n\t\t\t\t\t\t\tXx NO ITEMS TO REMOVE
DISCOUNT xX\n");
                else
                {
                        showProducts(aSBag20, nBagIndex, -1, -1);       //
shows all your items
                        printf("\n\t\t\t\t+ - - - - - - - - - - - - - - - - DISCOUNT:
REMOVE - - - - - - - - - - - - - - - - +\n");
                        removeDiscount(aSBag20, nBagIndex, nUID);
                }
                break;
        case '0':
                fflush(stdin);
```

```
                                                printf("\n\t\t\t\t\t\t    . . . EXIT FROM SELL MENU . . .\n");
                                                nBagIndex = 0;
                                                break;
                                    default:
                                                printf("\n\t\t\t\t\t\t\t Xx INVALID INPUT xX\n");
                        }
                }while (cSChoice != '0');
                break;
        case '2':
                do
                {
                        fflush(stdin);
                        printf("\n\t\t\t\t\t\t>>------------------------------------------------>>");
                        printf("\n\t\t\t\t\t\t|         PURCHASES FROM SELECTED ITEMS           |");
                        printf("\n\t\t\t\t\t\t|           ARE ENTITLED TO DISCOUNTS             |");
                        printf("\n\t\t\t\t\t\t>>------------------------------------------------>>\n");

                        printf("\n\t\t\t\t\t\t+ - - - - - - - - ACTION: B  U  Y - - - - - - - - +\n");
                        cBChoice = BuyMenu();
                        switch(cBChoice)
                        {
                                case '1':       if(nUSIndex == 0)
                                                        printf("\n\t\t\t\t\t\t   Xx THERE ARE CURRENTLY NO
PRODUCTS TO SHOW xX\n");

                                                else
                                                {
```

```
                                        printf("\n\t\t\t= = = = = = = = = = = = = = = = VIEW
ALL  PRODUCTS = = = = = = = = = = = = = = = = =\n");
                                        allProducts(aSItems, nUSIndex, nUID, aUserList,
nUIndex);
                                    }

                                    break;
                        case '2':       if(nUSIndex == 0)
                                        printf("\n\t\t\t\t\t\t   Xx THERE ARE CURRENTLY NO
PRODUCTS TO SHOW xX\n");
                                    else
                                    {
                                        printf("\n\t\t\t= = = = = = = = = = = = = = VIEW
DISCOUNTED  PRODUCTS = = = = = = = = = = = = = = =\n");
                                        viewDiscount(aSItems, nUSIndex, nUID, aUserList,
nUIndex);
                                    }

                                    break;
                        case '3':
                                    printf("\n\t\t\t= = = = = = = = = = = = SHOW ALL PRODUCTS
BY A SPECIFIC SELLER = = = = = = = = = = = =\n");
                                    specificSeller(aSItems, nUSIndex, nUID);
                                    break;
                        case '4':
                                    printf("\n\t\t\t= = = = = = = = = = = = = = SEARCH PRODUCTS
BY CATEGORY = = = = = = = = = = = = = = = =\n");
                                    searchCategory(aSItems, nUSIndex, nUID);
                                    break;
```

```
case '5':
                    printf("\n\t\t\t= = = = = = = = = = = = = = = SEARCH
PRODUCTS BY NAME = = = = = = = = = = = = = = =\n");
                    searchName(aSItems, nUSIndex, nUID);
                    break;
case '6':
                    printf("\n\t\t\t\t\t= = = = = = = = = ADD TO CART = = = = = = =
= =\n");
                    aCartList[nCartIndex] = addCart(aCartList, &nCartIndex,
aSItems, nUSIndex, nUID);
                    break;
case '7':
                    if(nCartIndex == 0)
                            printf("\n\t\t\t\t\t\t\tXx EMPTY CART xX\n");
                    else
                    {
                            do
                            {
                                    if(nCartIndex != 0)
                                    {
                                            printf("\n\n");
                                            for(x = 0; x < 77; x++)
                                                    printf("- ");
                                            printf("\n");
                                            displayCart(aCartList, nCartIndex);
```

```
}

printf("\n\t\t\t\t\t+ - - - - - - - - - EDIT   CART -
- - - - - - - - +\n");

cECChoice = editCart();
switch(cECChoice)
{
        case '1':

    if(nCartIndex == 0)

    printf("\n\t\t\t\t\t\t\tXx NO ITEMS IN CART xX\n");
                        else
                        {

    printf("\n\t\t\t\t\t~ ~ ~ ~ ~ ~ ~ REMOVE ALL SELLER ITEMS ~ ~ ~ ~ ~ ~ ~\n");

    removeSeller(aCartList, &nCartIndex);
                        }
                        break;
        case '2':

    if(nCartIndex == 0)

    printf("\n\t\t\t\t\t\t\tXx NO ITEMS IN CART xX\n");
                        else
                        {
```

```
printf("\n\t\t\t\t\t~ ~ ~ ~ ~ ~ ~ REMOVE  SPECIFIC  ITEM ~ ~ ~ ~ ~ ~ ~\n");

removeItem(aCartList, &nCartIndex, -1);
                                                                                                        }
                                                                                                        break;
                                                                                           case '3':

if(nCartIndex == 0)

printf("\n\t\t\t\t\t\t\tXx NO ITEMS IN CART xX\n");
                                                                                                        else
                                                                                                        {

printf("\n\t\t\t\t\t~ ~ ~ ~ ~ ~ ~ ~ ~ EDIT QUANTITY ~ ~ ~ ~ ~ ~ ~ ~ ~ ~\n");

editQty(aCartList, &nCartIndex, aSItems, nUSIndex);
                                                                                                        }
                                                                                                        break;
                                                                                           case '0':

printf("\n\t\t\t\t\t\t\t    . . . . FINISH EDIT CART . . . .\n");
                                                                                                        break;
                                                                                           default:

printf("\n\t\t\t\t\t\t\t\t Xx INVALID INPUT xX\n");
```

```
                                        }


                                }while(cECChoice != '0');
                        }


                        break;
                case '8':
                        if(nCartIndex == 0)
                                printf("\n\t\t\t\t\t\t\tXx EMPTY CART xX\n");
                        else
                        {
                                TransacList = confirmDate(TransacList);
                                if(TransacList.tD.nMonth > 0)
                                {
                                        TransacList.tIndex = 0;

                                        compareItem(aCartList, nCartIndex, aSItems,
nUSIndex);      // compares the item in the cart and in the list if there are changes

                                        do
                                        {
                                                printf("\n\t\t\t\t\t\t Entered Date :
%2d / %2d / %2d\n", TransacList.tD.nMonth, TransacList.tD.nDay, TransacList.tD.nYear);

                                                printf("\t\t\t\t\t+ - - - - - - - - - -
CHECK OUT - - - - - - - - - +\n");

                                                cCheckOut = checkOut();
                                                switch(cCheckOut)
```

```
                                                                                                    {
                                                                                                         case '1':

if(nCartIndex == 0)

printf("\n\t\t\t\t\t\t\tXx EMPTY CART xX\n");
                                                                                                                       else
                                                                                                                       {

printf("\n\t\t\t\t\t\t- - - - - - - - - - - - - - - - - A L L - - - - - - - - - - - - - - - - -\n");

i = 0;

while(i < nCartIndex)      // if the cart is greater than 0 it will do the following

{

     TransacList = transacSeller(aCartList, nCartIndex, aSItems, nUSIndex, TransacList, aCartList[i].cItem.sID);


     if(TransacList.tIndex > 0)

     {

          nCartIndex -= TransacList.tIndex;            // minus the buyers cart index

          TransacList = completeInfo(aUserList, nUIndex, TransacList, nUID);          // to copy the needed information to the transaction
```

```
                    saveTransac(TransacList);


                    displayReceipt(TransacList);

                    printf("\n\t\t\t\t\t\t\t\t\t<< END >>\n\n\n");

          }


          for(j = 0; j < nUSIndex; j++) // to check if quantity is greater than the item index

                    if(aSItems[j].iID == aCartList[i].cItem.iID)

                              if(aCartList[i].cQty > aSItems[j].iQty || aCartList[i].cQty == 0)

                                        i++;

}
                                                                                                                                }



break;
                                                                                                              case '2':

if(nCartIndex == 0)

printf("\n\t\t\t\t\t\t\tXx EMPTY CART xX\n");
                                                                                                                                else
```

```
                                                                                                                                                    {

printf("\n\t\t\t\t\t- - - - - - - - - - - - - - - - - SPECIFIC SELLER - - - - - - - - - - - - - - - - -\n");



nCheck = 1;

x = -1;

do

{

        TransacList = transacSeller(aCartList, nCartIndex, aSItems, nUSIndex, TransacList, x);



        if(TransacList.tIndex != 0)          // if the transac index is not equal to 0 it will do the following

        {

                nCartIndex -= TransacList.tIndex;           // minus the buyers cart index

                TransacList = completeInfo(aUserList, nUIndex, TransacList, nUID);          // to copy the needed information for the transaction

                saveTransac(TransacList);

                displayReceipt(TransacList);
```

```
			printf("\n\t\t\t\t\t\t\t\t\t<< END >>\n\n\n");

		}

		x = TransacList.sId;		// gets the seller id

		if(checkCSellerID(aCartList, nCartIndex, x) == 1)  // if found

		{

			for(i = 0; i < nCartIndex; i++)		// loop for the items in cart

			{

				if(x == aCartList[i].cItem.sID)		// if seller id is the same to the items seller in the cart

				{

					for(j = 0; j < nUSIndex; j++)		// loop for the list of users

					{

						if(aSItems[j].iID == aCartList[i].cItem.iID)// if seller id is the same in the list of items and the cart list of items

						{

							if(aCartList[i].cQty <= aSItems[j].iQty && aCartList[i].cQty > 0)	// if the quantity in the cart is greater than 0 or less than or equal to the
available quantity

							{
```

```
                            nCheck = 1;

                            j = nUSIndex;     // stop the loop for list of users

                            i = nCartIndex;   // stop the loop for list of cart items

                        }

                        else

                            nCheck = 0;

                    }

                }

            }

        }

    }

    else

        nCheck = 0;

}while(nCheck == 1);
                                                                                            }

break;
```

```
                                                                                                                        case '3':

        if(nCartIndex == 0)

        printf("\n\t\t\t\t\t\t\tXx EMPTY CART xX\n");
                                                                                                                                        else

                                                                                                                                        {



        printf("\n\t\t\t\t\t- - - - - - - - - - - - - - - -  SPECIFIC ITEM  - - - - - - - - - - - - - - - -\n");

        TransacList = transacItem(aCartList, nCartIndex, aSItems, nUSIndex, TransacList);

        if(TransacList.tIndex != 0)

        {

                nCartIndex -= TransacList.tIndex;          // minus the buyers cart index

                TransacList = completeInfo(aUserList, nUIndex, TransacList, nUID);         // to copy the needed information for the transaction

                saveTransac(TransacList);


                displayReceipt(TransacList);

                printf("\n\t\t\t\t\t\t\t\t\t<< END >>\n\n\n");

        }
                                                                                                                                        }
```

```
            break;
                                                                                                        case '0':
            printf("\n\t\t\t\t\t\t\t    . . . EXIT FROM CHECK OUT . . .\n");
            break;
                                                                                                        default:
            printf("\n\t\t\t\t\t\t\t\t Xx INVALID INPUT xX\n");
                                                                                                   }
                                                                                          }while(cCheckOut != '0');
                                                                                     }
                                                                                }
                                                                                break;
                                                                  case '9':
                                                                                if(nCartIndex == 0)
                                                                                     printf("\n\t\t\t\t\t\t\t\tXx EMPTY CART xX\n");
                                                                                else
                                                                                {
                                                                                     printf("\n+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
DISPLAY / COMPARE  ITEMs  IN  CART - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +");
                                                                                     compareItem(aCartList, nCartIndex, aSItems,
nUSIndex);        // compares the item in the cart and in the list if there are changes
                                                                                     if(nCartIndex != 0)
                                                                                          displayCart(aCartList, nCartIndex);
```

```
                                }
                                break;
                        case '0':
                                printf("\n\t\t\t\t\t\t\t    . . . EXIT FROM BUY MENU . . .\n");
                                saveCart(aCartList, nCartIndex, nUID);
                                break;
                        default:
                                printf("\n\t\t\t\t\t\t\t\t Xx INVALID INPUT xX\n");
                    }
            }while (cBChoice != '0');
            break;
    case '3':
            do
            {
                    fflush(stdin);
                    printf("\n\t\t\t\t\t\t+ - - - - - - - SHOW TRANSACTIONS - - - - - - - - +\n");
                    cSTChoice = showTransacMenu();
                    switch(cSTChoice)
                    {
                        case '1':
                                printf("\n\t\t- - - - - - - - - - - - - - - - - - - - - - -  YOUR  SOLD
PRODUCTS  - - - - - - - - - - - - - - - - - - - - - - -\n");

                                adminSellerSales(aUser.ID, aUser.Name);
                                break;
                        case '2':
```

```
                                                                        printf("\n\t\t- - - - - - - - - - - - - - - - - - - - -  YOUR  BOUGHT
PRODUCTS  - - - - - - - - - - - - - - - - - - - - - -\n");

                                                                        userReceipt(nUID);

                                                                        break;

                                                            case '0':

                                                                        printf("\n\t\t\t\t\t\t. . . EXIT FROM SHOW TRANSACTIONS . .
.\n");

                                                                        break;

                                                            default:

                                                                        printf("\n\t\t\t\t\t\t\t Xx INVALID INPUT xX\n");

                                                        }

                                                    }while(cSTChoice != '0');

                                                    break;

                                        case '0':

                                                    fflush(stdin);

                                                    printf("\n\t\t\t\t\t\t    . . . EXIT FROM ACTIONS . . .\n");

                                                    break;

                                        default:

                                                    printf("\n\t\t\t\t\t\t\t Xx INVALID INPUT xX\n");

                                    }

                                }while(cUAction != '0');

                            }

                            break;

                case '3'://system ("cls");

                            printf("\n\t\t\t\t\t Enter Admin Password ");

                            printf("\n\t\t\t\t\t Password :: ");
```

```
		scanf("%s", cAdmin);
		if(strcmp(cAdmin, "H3LLo?") != 0)
			printf("\n\t\t\t\t\t\t  Xx UNAUTHORIZED ACCESS NOT ALLOWED xX\n");
		else
		{
			do
			{
				printf("\n\t\t\t\t\t\t+ - - - - - -  A D M I N   M E N U  - - - - - - - +\n");
				cAChoice = AdminMenu();
				fflush(stdin);
				switch(cAChoice)
				{
					case '1':
							if (nUIndex == 0)
								printf("\n\t\t\t\t\t\t   Xx NO USERS TO DISPLAY xX\n");
							else
							{
								printf("\n- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - SHOW ALL USERS - - - - - - - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - -\n");

								adminUsers(aUserList, nUIndex);
							}
							fflush(stdin);
							break;
					case '2':
							if (nUSIndex == 0)
								printf("\n\t\t\t\t\t\t   Xx NO SELLERS TO DISPLAY xX\n");
```

```
                        else
                        {
                                printf("\n----------------------------------- SHOW ALL SELLERS -------------------------------------\n");

                                adminSellers(aSItems, nUSIndex, aUserList, nUIndex);
                        }
                        fflush(stdin);
                        break;
                case '3':
                        fflush(stdin);
                        printf("\n\t\t\t-------------- SHOW TOTAL SALES IN GIVEN DURATION -------------\n");
                        adminTotalSales();
                        break;
                case '4':
                        fflush(stdin);
                        printf("\n\t\t\t-----------------------  SHOW  SELLER  SALES  -----------------------\n");
                        adminSellerSales(-1, NULL);
                        break;
                case '5':
                        fflush(stdin);
                        printf("\n\t\t\t------------------------  SHOW  SHOPAHOLICS  -----------------------\n");
                        adminShopaholics(aUserList, nUIndex);
                        break;
                case '6':
                        fflush(stdin);
```

```
                                        printf("\n\t\t- - - - - - - - - - - - - - - - - - - SHOW ALL TRANSACTIONS BY SPECIFIC SELLER - - - - - - - - - - - - - - 
- - - -\n");

                                        printf("\n\t\t\tENTER SELLER ID :: ");
                                        scanf("%d", &x);
                                        nCheck = 0;
                                        for(i = 0; i < nUIndex; i++)
                                                if(x == aUserList[i].ID)
                                                        nCheck = 1;
                                        if(nCheck == 0)
                                                printf("\n\t\t\t\t\t\t\t   Xx USER ID NOT FOUND xX\n");
                                        for(i = 0; i < nUIndex && nCheck == 1; i++)
                                        {
                                                if(x == aUserList[i].ID)
                                                {
                                                        printf("\n\t\t\tTRANSACTIONS OF %s, %d\n", aUserList[i].Name, aUserList[i].ID);

                                                        adminSellerSales(aUserList[i].ID, aUserList[i].Name);
                                                }
                                        }
                                        break;
                                case '7':
                                        fflush(stdin);
                                        printf("\n\t\t- - - - - - - - - - - - - - - - - - - SHOW ALL TRANSACTIONS BY SPECIFIC BUYER - - - - - - - - - - - - - - 
- - - -\n");

                                        printf("\n\t\t\tENTER BUYER ID :: ");
                                        scanf("%d", &x);
                                        nCheck = 0;
```

```
                                    for(i = 0; i < nUIndex; i++)
                                        if(x == aUserList[i].ID)
                                            nCheck = 1;
                                    if(nCheck == 0)
                                        printf("\n\t\t\t\t\t\t Xx USER ID NOT FOUND\n");
                                    for(i = 0; i < nUIndex && nCheck == 1; i++)
                                    {
                                        if(x == aUserList[i].ID)
                                        {
                                            printf("\n\t\t\tTRANSACTIONS OF %s, %d\n", aUserList[i].Name, aUserList[i].ID);

                                            userReceipt(aUserList[i].ID);
                                        }
                                    }
                                    break;
                        case '0':
                                    fflush(stdin);
                                    printf("\n\t\t\t\t\t\t    . . . EXIT FROM ADMIN MENU . . .\n");
                                    break;
                        default:
                                    printf("\n\t\t\t\t\t\t\t Xx INVALID INPUT xX\n");
                    }
                }while(cAChoice != '0');
            }
            break;
case '0':
```

```
                        if(0 < nUIndex)
                        {
                                saveUsers(aUserList, nUIndex);
                                saveItems(aSItems, nUSIndex);
                        }
                        printf("\n\t\t\t\t\t\t\t>>-- D O N E --<<\n");
                        break;
            default:
                        printf("\n\t\t\t\t\t\t\t Xx INVALID INPUT xX\n");
        }
    }while(cMMChoice != '0');
    return 0;
}
```