**SOURCE CODE**

//INCLUDING THE NECESSARY HEADER FILES

//FOR BASIC INPUT OUTPUT FUNCTIONS(NOT COMPULSORY AS FSTREAM IS USED)

#include<iostream.h>

//FOR FUNCTIONS LIKE CLEAR SCREEN , GETTING CHARACTERS

#include<conio.h>

//FOR FUNCTIONS LIKE GETS

#include<stdio.h>

//FOR EXIT FUNCTIONS

#include<process.h>

//FOR DATA FILE HANDLING PURPOSES

#include<fstream.h>

//FOR USING STRCPY,STRCMPI AND OTHER STRING OPERATIONS

#include<string.h>

//FOR CALCULATION PURPOSES

#include<math.h>

// FOR USING RAND,SRAND

#include<stdlib.h>

//FOR USING OF TIME VARIABLES

#include<time.h>

//DECLARING THE CLASS

class product

{

private://PRIVATE MEMBERS

int pno; //PRODUCT NUMBER

char name[1000]; //PRODUCT NAME

float price, qty, gst, dis; //DECLARING PRICE,GST,DISCOUNT

int quantity; //DECLARING QUANTITY

public: //PUBLIC MEMBERS

void create\_product() //FUNCTION TO CREATE A NEW PRODUCT

{

char ch;

cout<<"\nPlease Enter The Product No. : ";

cin>>pno;

cout<<"\n\t\t\t"<<endl;

cout<<"PRESS Y IF YOU WANT TO ENTER THE PRODUCT NO. AGAIN\n\t\t\tELSE PRESS N"<<endl;

cin>>ch;

if(ch=='Y'||ch=='y')

{

cin>>pno;

}

cout<<"\n\nPlease Enter The Name Of The Product : ";

gets(name);

cout<<"\nPlease Enter The Price Of The Product(Rs) : ";

cin>>price;

cout<<"\nPlease Enter The Discount (%) : ";

cin>>dis;

cout<<"\nPlease Enter The Quantity : ";

cin>>quantity;

}

void show\_product() //FUNCTION TO SHOW THE DETAILS OF THE PRODUCT

{

cout<<"\n Product No. : "<<pno;

cout<<"\n Name of The Product : ";

puts(name);

cout<<"\n Price of The Product : "<<price<<" Rs";

cout<<"\n Discount : "<<dis<<"%";

}

int retpno() //FUNCTION TO RETURN PRODUCT NO.

{

return pno;

}

float retprice() //FUNCTION TO RETURN PRODUCT PRICE

{

return price;

}

char\* retname() //FUNCTION TO RETURN PRODUCT NAME

{

return name;

}

float retdis() //FUNCTION TO RETURN PRODUCT DISCOUNT

{

return dis;

}

int retquantity() //FUNCTION TO RETURN PRODUCT QUANTITY

{

return quantity;

}

//DECLARING MEMBER FUNCTIONS

void purchase();

void defect();

int billing(int, int);

void mod\_product(int);

};

//CLASS ENDS HERE

// GLOBAL DECLARTION FOR STREAM OBJECT AND OBJECT OF CLASS

fstream fp; //FOR Shop.dat

product pr; //CLASS OBJECT

fstream ep;//FOR employee.dat

fstream bp;//FOR balance.dat

//DEFINITION OF MEMBER FUNCTIONS OF CLASS PRODUCT

void product::purchase() //FOR INCREASING PRODUCT QUANTITY

{

//OPENING SHOP.DAT

fp.open("Shop.dat",ios::in|ios::out|ios::binary);

//DECLARING REQUIRED VARIBALES

int no,flag=0;

long pos; //FOR POSITION OF CURSOR

int quantity\_d;

cout<<"Enter Product No. For Which Quantity Is To Be Updated : "<<endl;

cin>>no;

cout<<"Enter Quantity : "<<endl;

cin>>quantity\_d;

while(!fp.eof()) //RUNS UNTIL END OF SHOP.DAT IS REACHED

{

pos=fp.tellg(); //GIVES THE CURRENT POSITION OF THE CURSOR

fp.read((char \*)&pr,sizeof(product)); //FOR READING DATA FROM FILE

if(pr.retpno()==no)

{

strcpy(name,name);

pno=pno;

dis=dis;

price=price;

quantity=quantity+quantity\_d;

fp.seekg(pos); //SETTING THE POSITION OF THE POINTER TO OVERWRITE THE DATA

fp.write((char \* )&pr,sizeof(product)); //WRITING IN FILE

flag=1;

cout<<"\n\n==============================================================================\n\t\t\tQuantity Updated successfully!!!!!";

break;

}

}

fp.close(); //CLOSING SHOP.DAT

if(flag==0)

{

cout<<"Invalid Product No....."<<endl;

}

}

//FOR DECREASING PRODUCT QUANTITY

void product::defect()

{

fp.open("Shop.dat",ios::in|ios::out|ios::binary); //OPENING SHOP.DAT

//DECLARING VARIABLES

int no,flag=0;

long pos;

int quantity\_d;

cout<<"Enter Product No. For Which Quantity Is To Be Returned : "<<endl;

cin>>no;

cout<<"Enter Quantity : "<<endl;

cin>>quantity\_d;

while(!fp.eof())

{

pos=fp.tellg();

fp.read((char \*)&pr,sizeof(product));

if(pr.retpno()==no &&(quantity-quantity\_d)>=0)

{

strcpy(name,name);

pno=pno;

dis=dis;

price=price;

quantity=quantity-quantity\_d;

fp.seekg(pos);

fp.write((char \* )&pr,sizeof(product));

flag=1;

cout<<"Quantity Updated Successfully";

break;

}

}

fp.close();

if(flag==0)

{

cout<<"Invalid Input........"<<endl;

}

}

//FOR CHECKING AND UPDATING QUANTITY DURING BILLING

int product::billing(int no, int quantity\_d) //PARAMETERS PASSED FOR PRODUCT NO AND QUANTITY FROM USER

{

fp.open("Shop.dat",ios::in|ios::out|ios::binary); //OPENING SHOP.DAT

int flag=0;

long pos;

while(!fp.eof())

{

pos=fp.tellg();

fp.read((char \*)&pr,sizeof(product));

if(pr.retpno()==no && (quantity-quantity\_d)>=0)

{

strcpy(name,name);

pno=pno;

dis=dis;

price=price;

quantity=quantity-quantity\_d;

fp.seekg(pos);

fp.write((char \* )&pr,sizeof(product));

flag=1;

break;

}

}

fp.close();

if(flag==0)

{

cout<<"Invalid Input......."<<endl;

}

return flag;

}

//END OF DEFINITION OF MEMBER FUNCTIONS

void product::mod\_product(int pno)

{

pno=pno;

cout<<"\n\nPlease Enter The Name Of The Product : ";

gets(name);

cout<<"\nPlease Enter The Price Of The Product(Rs) : ";

cin>>price;

cout<<"\nPlease Enter The Discount (%) : ";

cin>>dis;

cout<<"\nPlease Enter The Quantity : ";

cin>>quantity;

}

//UDF FOR ADMIN USERNAME AND PASSWORD(SIGNING INTO ADMIN PANEL)

int password()

{

struct id

{

char name[10];

char pass[5];

}ids[3];

clrscr();

char u\_name[10];

char pass\_w[4];

int x,flag=0,y,flag2=0;

//PRECODED IDS AND PASSWORDS

strcpy(ids[0].name,"Sanskar");

strcpy(ids[1].name,"Shivam");

strcpy(ids[2].name,"Vaishali");

strcpy(ids[0].pass,"2002");

strcpy(ids[1].pass,"2001");

strcpy(ids[2].pass, "2107");

gotoxy(31,11);

cout<<"Enter Username : ";

gets(u\_name);

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

{

x=strcmpi(u\_name,ids[i].name);

if(x==0)

{

flag=1;

break;

}

}

if(flag==0)

{

clrscr();

gotoxy(33,11);

cout<<"Wrong Username......."<<endl;

getch();

}

else if(x==0)

{

gotoxy(33,13);

cout<<"Enter Password : ";

for(int j=0;j<4;j++)

{

pass\_w[j]=getch();

cout<<"\*";

}

cout<<endl;

for(int x=0;x<4;x++)

{

flag2=0;

if(pass\_w[x]==ids[i].pass[x])

{

flag2=1;

}

}

if(flag2==1)

{

clrscr();

gotoxy(33,13);

cout<<"Sign in Successfull!!!!!"<<endl;

}

else

{

clrscr();

gotoxy(33,13);

cout<<"Sign in Failed......."<<endl;

}

getch();

}

return flag2;

}

//UDF FOR CREATING EMPLOYEE

void create\_employee()

{

int emp\_no,empx,y;

cout<<"Enter The Employee No. To Be Created(NUMERICS ONLY) : "<<endl;

ep.open("employee.dat",ios::app); //OPENING EMPLOYEE.DAT

ep.seekg(0,ios::beg); //SETTING THE CURSOR POSITION TO 0th BYTE FROM BEGINNING

cin>>emp\_no;

ep<<emp\_no<<endl;

ep.close();

cout<<"Employee Id Created Successfully!!!!!";

}

//FOR SIGNING INTO EMPLOYEE PANEL

int user\_id()

{

clrscr();

int emp\_no1,emp\_no,flag=0;

cout<<"Enter Your Employee No : ";

cin>>emp\_no1;

ep.open("employee.dat",ios::in); //OPENING THE FILE

ep.seekg(0);

while(!ep.eof())

{

ep>>emp\_no;

if(emp\_no==emp\_no1)

{

cout<<"Sign In Succesful!!!!!"<<endl;

flag =1;

getch();

break;

}

}

if(flag==0)

{

cout<<"Sign In Failed......"<<endl;

getch();

}

ep.close();

return flag;

}

//FUNCTION TO WRITE IN FILE SHOP.DAT

void write\_product()

{

int flag=0, no;

cout<<"Please Enter The Product No. To Check Whether It Exists : "<<endl;

cin>>no;

fp.open("Shop.dat",ios::in|ios::out|ios::app); /\*CREATING AND OPENING

FILE NAMED SHOP.DAT AND APPENDING

IT SO THAT DATA DOES NOT GET

OVERWRITTEN\*/

while(fp.read((char\*)&pr,sizeof(product)))

{

if(pr.retpno()==no)

{

flag=1;

break;

}

}

fp.close();

if(flag==0)

{

fp.open("Shop.dat",ios::out|ios::app);

pr.create\_product(); //CALLING FUNCTION FOR CREATING PRODUCT

fp.write((char\*)&pr,sizeof(product)); //WRITING IN FILE

fp.close();

cout<<"\n\n======================================\nThe Product Has Been Created!!! ";

getch();

}

if(flag==1)

{

cout<<"Product Already Exists!!!"<<endl;

getch();

}

}

// FUNCTION TO READ ALL RECORS FROM FILE

void display\_all()

{

clrscr();

cout<<"\n\n\n\t\tDISPLAYING ALL RECORDS !!!\n\n";

fp.open("Shop.dat",ios::in); //OPENING FILE

while(fp.read((char\*)&pr,sizeof(product)))

{

pr.show\_product(); //CALLING FUNCTION TO DISPLAY PRODUCTS

cout<<"\n\n================================================================================\n";

getch();

}

fp.close(); //CLOSING FILE

getch();

}

// FUNCTION TO READ SPECIFIC RECORD FROM FILE

void display\_sp(int n)

{

int flag=0;

fp.open("Shop.dat",ios::in); //OPENING FILE

while(fp.read((char\*)&pr,sizeof(product)))

{

if(pr.retpno()==n)

{

clrscr();

pr.show\_product(); //CALLING FUNCTION TO DISPLAY PRODUCT

flag=1;

break;

}

}

fp.close(); //CLOSING FILE

if(flag==0) //IF PRODUCT IS NOT IN RECORD (DIRECTORY)

{

cout<<"\n\nRecord Does Not Exist";

}

getch();

}

//FUNCTION TO MODIFY THE RECORD

void modify\_product()

{

int no,found=0;

clrscr();

cout<<"\n\n\tTo Modify, ";

cout<<"\n\n\tPlease Enter The Product No. : ";

cin>>no;

fp.open("Shop.dat",ios::in|ios::out); //OPENING THE FILE

while(fp.read((char\*)&pr,sizeof(product)) && found==0) //READING THE FILE

{

if(pr.retpno()==no)

{

pr.show\_product(); //FUNCTION TO SHOW THE ORIGINAL DETAILS OF THE PRODUCT

cout<<"\n=====================================================\nPlease Enter The New Details "<<endl;

pr.mod\_product(no); //CALLING A FUNCTION TO CREATE A NEW PRODUCT

int pos=-1\*sizeof(pr);

fp.seekp(pos,ios::cur); //BACKING UP THE POINTER FROM CURRENT POSITION

fp.write((char\*)&pr,sizeof(product)); //OVERWRITING THE DESIRED RECORD WITH THE NEW USER ENTERED DETAILS

cout<<"\n======================================================\n\t\t Record Updated Successfully";

found=1;

}

}

fp.close(); //CLOSING THE FILE

//IF NO PRODUCT IS FOUND

if(found==0)

{

cout<<"\n\n Record Not Found!!!!! ";

}

getch();

}

//FUNCTION TO DELETE A SPECIFIC PRODUCT

void delete\_product()

{

int no;

clrscr();

cout<<"\n\n\n\t To Delete Record,";

cout<<"\n\nPlease Enter The Product (No.) You Want To Delete : ";

cin>>no;

fp.open("Shop.dat",ios::in|ios::out); //OPENING FILE

fstream fp2; //DECLARING A TEMPORARY STREAM FILE

fp2.open("Temp.dat",ios::out); //OPENING THE TEMPORARY FILE

fp.seekg(0,ios::beg);

/\*THE BELOW SEGMENT COPIES ALL THE PRODUCT - RECORDS TO THE OTHER FILE

OTHER THAN THE ONE TO BE DELETED\*/

while(fp.read((char\*)&pr,sizeof(product)))

{

if(pr.retpno()!=no)

{

/\*

WRITING ALL OF THE RECORDS INTO THE TEMPORARY FILE

EXCEPT THE SPECIFIC RECORD (WHICH THE USER WANT TO BE DELETED\*/

fp2.write((char\*)&pr,sizeof(product));

}

}

fp2.close(); //CLOSING FILE

fp.close(); //CLOSING FILE

remove("Shop.dat"); //DELETING SHOP FILE WHICH CONTAINS ALL PRODUCTS

rename("Temp.dat","Shop.dat"); //RENAMING TEMPORARY FILE AS SHOP.DAT(MAIN FILE)

cout<<"\n\n\t\tRecord Deleted .......";

getch();

}

//FUNCTION TO DISPLAY INVENTORY

void displayinventory()

{

clrscr();

fp.open("Shop.dat",ios::in); //OPENING FILE

if(!fp) //IF FILE FAILED TO OPEN(TO PREVENT A MEMORY LEAK)

{

cout<<"ERROR!!! FILE COULD NOT BE OPEN\n\n\n Go To Admin Menu to create File";

cout<<"\n\n\n Program is closing ....";

getch();

exit(0);

}

//LAYOUT OF THE INVENTORY SCREEN

cout<<"\n\n\t\t INVENTORY \n\n";

cout<<"=============================================================================\n";

cout<<"P.NO.\t\tNAME\t\t\t\tQUANTITY\n";

cout<<"=============================================================================\n";

int i=0;

while(fp.read((char\*)&pr,sizeof(product)) && i<=9)

{

/\*DISPLAYING ALL THE VALUES(OF THEIR CORRESPONDING PRODUCTS)

BY CALLING THE PUBLIC MEMBER FUNCTIONS \*/

cout<<pr.retpno()<<"\t\t"<<pr.retname()<<"\t\t\t "<<pr.retquantity()<<endl;

if(i==9)

{

i=0;

getch();

}

else

{

i++;

}

}

fp.close(); //CLOSING FILE

}

// FUNCTION TO PLACE ORDERS AND GENERATE BILLS ACCORDINGLY

void place\_order()

{

int order\_arr[50],quan[50],c=0;

long amt,damt=0;

long gamt, total=0;

char ch='Y';

displayinventory(); //FUNCTION TO SHOW THE INVENTORY

int check=0;

cout<<"\n============================";

cout<<"\n PLACE YOUR ORDER";

cout<<"\n============================\n";

do

{

cout<<"\n\nEnter The Product No. : ";

cin>>order\_arr[c];

cout<<"\nQuantity : ";

cin>>quan[c];

//FUNCTION 'BILLING' RETURNS VALUE ACCORDINGLY IF A

//VALID ORDER CAN BE PLACED BY THE DATA ENTERED BY THE USER

check= pr.billing(order\_arr[c], quan[c]);

if(check==1)

{

c++;

}

cout<<"\nDo You Want To Order Another Product ? (Y/N)";

cin>>ch;

}while(ch=='y' ||ch=='Y');

cout<<"\n\nThank You For Placing The Order";

getch();

clrscr();

cout<<"\n";

srand(time(0)); //SEEDING(INITIALIZING) SRAND(IN TURN TO RAND() ) TO THE CURRENT TIME

int order\_no=rand();

//LAYOUT OF INVOICE

"\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*INVOICE\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout<<"\nOrder no."<< order\_no<<endl;

/\* THIS NEXT SEGMENT IS USED TO DISPLAY THE CURRENT TIME & DATE

ON THE INVOICE \*/

time\_t tt; //DECLARING A TIME VARIABLE

struct tm\*ti;

time(&tt);

ti=localtime(&tt);

cout<<"\n"<<asctime(ti)<<endl<<endl;

cout<<"==============================================================================";

cout<<"\nPr No.\tName\t\tQuantity\tPrice\t\tAmount\t GST Amount\n";

cout<<"==============================================================================\n";

//RUNNING A LOOP TILL THE TOTAL NO. OF DIFFERENT PRODUCTS ORDERED

for(int x=0;x<c;x++)

{

fp.open("Shop.dat",ios::in); //OPENING FILE(SHOP.DAT)

fp.read((char\*)&pr,sizeof(product));

while(!fp.eof()) //LOOP WHICH RUNS TILL THE END OF SHOP.DAT

{

if(pr.retpno()==order\_arr[x])

{

//STORING AMOUNT IN AMT

amt=pr.retprice()\*quan[x];

//STORING DISCOUNT AMOUNT IN DAMT

damt=amt-(amt\*pr.retdis()/100);

//STORING GST(TAX)AMOUNT IN GAMT;

gamt=(amt\*0.18);

//GETING THE TOTAL AMOUNT IN TOTAL

total=total+gamt+damt;

bp.open("Balance.dat",ios::app);//OPENING FILE

bp.seekg(0,ios::beg);

//WRITING THE ORDER NOS. GENERATED AND TOTAL AMOUNT INTO BALANCE.DAT

//FOR BALANCE SHEET

bp<<order\_no<<" "<<total<<endl;

bp.close();

//DISPLAYING ALL THE VALUES ONTO THE INVOICE

cout<<"\n"<<order\_arr[x]<<"\t"<<pr.retname()

<<"\t"<<quan[x]<<"\t\t"<<pr.retprice()<<"\t\t"<<amt<<"Rs"<<" "<<gamt<<" Rs";

}

fp.read((char\*)&pr,sizeof(product));

}

fp.close();

}

//DISPLAYING DISCOUNTED AMOUNT

cout<<"\n\n\nDiscounted Amount = "<<damt<<" Rs"<<"\n";

cout<<"================================================================================\n\t\t\t\t GRAND TOTAL(Inc. of GST) = "<<total<<" Rs";

getch();

}

//FUNCTION FOR THE BALANCE SHEET

void balance\_sheet()

{

int order;

long amount, sale=0;

bp.open("Balance.dat", ios::in); //OPENING FILE

bp.seekg(0);

//LAYOUT OF THE BALANCE SHEET

cout<<"\n\n\t\t BALANCE SHEET \n\n";

cout<<"=============================================================================\n";

cout<<"ORDER NO.\t\t\tORDER AMOUNT\n\n";

cout<<"=============================================================================\n";

while(bp>>order>>amount) //RUNNING THE LOOP TILL THE END OF ALL THE RECORDS

{

sale=sale+amount;

//DISPLAYING ALL THE VALUES

cout<<order<<"\t\t\t\t"<<amount<<" Rs"<<endl;

}

bp.close();

cout<<"\n\n======================================================\nTotal sale is = "<<sale<<"Rs";

}

// INTRODUCTION FUNCTION FOR THE PROGRAM

void intro()

{

clrscr();

gotoxy(31,11);

cout<<"BEST BUY";

gotoxy(40,11);

cout<<"ELECTRONIC";

gotoxy(38,14);

cout<<"STORE";

getch();

}

// ADMINSTRATOR MENU FUNCTION

void admin\_menu()

{

clrscr();

int ch2;

//DISPLAYING ALL THE DIFFERENT CHOICES ON THE ADMIN MENU

cout<<"\n\n\tADMIN MENU";

cout<<"\n\n\t1.CREATE PRODUCT";

cout<<"\n\n\t2.DISPLAY ALL PRODUCTS";

cout<<"\n\n\t3.SEARCH PRODUCT ";

cout<<"\n\n\t4.MODIFY PRODUCT";

cout<<"\n\n\t5.DELETE PRODUCT";

cout<<"\n\n\t6.VIEW INVENTORY";

cout<<"\n\n\t7.EDIT INVENTORY";

cout<<"\n\n\t8.DEFECTED PRODUCT";

cout<<"\n\n\t9.BALANCE SHEET ";

cout<<"\n\n\t10.CREATE EMPLOYEE";

cout<<"\n\n\t11.BACK TO MAIN MENU";

cout<<"\n\n\tPlease Enter Your Choice (1-11) ";

cin>>ch2;

//PROCESSING ALL THE CASES(CHOICES) ACCORDINGLY

//CALLING ALL THE REQUIRED FUNCTIONS IN DIFFERENT CASES

switch(ch2)

{

case 1: clrscr();

write\_product();

admin\_menu();

break;

case 2: display\_all();

admin\_menu();

break;

case 3: clrscr();

int no;

cout<<"Enter product no."<<endl;

cin>>no;

display\_sp(no);

admin\_menu();

break;

case 4: modify\_product();

admin\_menu();

break;

case 5: delete\_product();

admin\_menu();

break;

case 6: displayinventory();

getch();

admin\_menu();

break;

case 11: break;

case 7 :clrscr();

pr.purchase();

getch();

admin\_menu();

break;

case 8 :clrscr();

pr.defect();

getch();

admin\_menu();

break;

case 10 :clrscr();

create\_employee();

getch();

admin\_menu();

break;

case 9:clrscr();

balance\_sheet();

getch();

admin\_menu();

break;

default: cout<<"\a";admin\_menu();

}

}

// STARTING OF THE MAIN FUNCTION OF PROGRAM

void main()

{

char ch;

//USING CUSTOM BACKGROUND AND FOREGROUND

intro();

do

{

clrscr();

//TO DISPLAY THE CURRENT TIME ON THE SCREEN

time\_t tt;

struct tm\*ti;

time(&tt);

ti=localtime(&tt);

cout<<asctime(ti)<<endl;

cout<<"\n\n\n\tMAIN MENU";

//DISPLAYING ALL THE CHOICES ON THE SCREEN

cout<<"\n\n\t01. EMPLOYEE";

cout<<"\n\n\t02. ADMINISTRATOR";

cout<<"\n\n\t03. EXIT";

cout<<"\n\n\t04. ABOUT";

cout<<"\n\n\tPlease Select Your Option (1-4) ";

//ACCEPTING THE INPUT

ch=getche();

switch(ch)

{

case '1': clrscr();

/\*REQUESTING EMPLOYEE NO. TO ALLOW THE USER TO ENTER THE

PLACE ORDER MENU \*/

int y=user\_id();

if(y==1)

{

place\_order();

}

break;

case '2': /\*REQUESTING USERNAME AND PASSWORD TO ALLOW THE USER TO ENTER THE

ADMIN MENU \*/

int x=password();

if(x==1)

{

admin\_menu();

}

break;

case '3'://TO EXIT THE PROGRAM

exit(0);

case '4':clrscr();

//TO SEE ABOUT SECTION OF THE PROGRAM

cout<<"\n\n\n\n\n\n\n\n\n\n\n\t\t\tMade by : Sanskar Gupta And Shivam Dargan"<<endl;

cout<<"\n\n\n\t\t\t\tBEST BUY BILLING SYSTEM\n\n\n\t\t\t\t\tVERSION 1.0.0"<<endl;

getch();

break;

default :cout<<"\a";

}

}while(ch!='3');

}

// END OF PROJECT