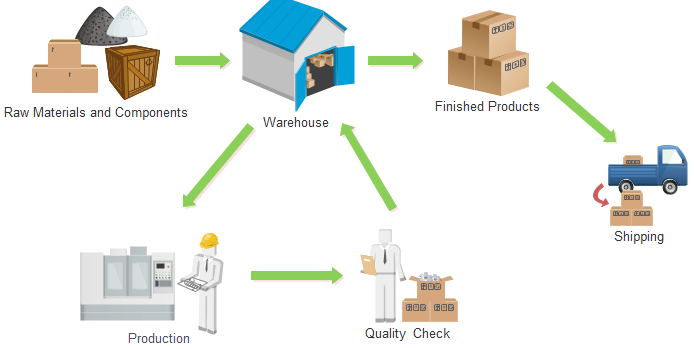
***Inventory Management System***

***One-page write-up***

An Inventory Management System (IMS) is a comprehensive solution that enables businesses to organize, track, ensuring businesses have the right amount of stock to meet customer demand while minimizing excess and associated costs and optimize their inventory efficiently.



**Key Components and Functionalities:**

**1) Item Master/product management:**

* An Item Master serves as a centralized repository for detailed information about each individual item within a business's inventory.
* It acts as a comprehensive catalog, providing essential details to streamline various inventory-related processes, enhance decision-making, and ensure efficient management of good,

**2) Party Master:**

* A Party Master, often referred to as a Customer or Vendor Master, provides a centralized repository for detailed information about individuals with whom a business engages in transactions with their unique Gst No.

**3) Stock Tracking:**

* Real time tracking of current stock levels if each item as on date.
* Provides insights into stock movement, allowing businesses to make data-driven decisions.

**4) Sales and Invoicing:**

* Integrates with point-of-sale systems or online platforms to automatically update inventory levels upon sales.
* Generates invoices and receipts, facilitating smooth transactions with customers.

**5) User Access Control:**

* Implements role-based access control to ensure that only authorized personnel can view, modify, or delete sensitive inventory data.

**Limitations**

* In this project we don’t have the functionality to give alert when the stock reaches the minimum level.
* Stock reports cann’t be generated with date constraints.

***Code***

**Main Menu**

#include<stdio.h>

#include<stdlib.h>

#include "entry.h"

#include<time.h>

struct report{

int itemcode;

int stock;

char itemname[100];

struct report \*loc;

};

struct report \*srep=NULL;

void masters();

void entry();

void report();

void masters();

void party\_master();

void item\_master();

void reportadd(struct report\*);

void reportprint();

int main()

{

while(1){

int ch;

printf("INVENTORY MANAGEMENT SYSTEM\n");

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

printf("Main Menu:\n");

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

printf("1.Entry\n");

printf("2.Report\n");

printf("3.Masters\n");

printf("4.Exit\n");

printf("Enter your choice:");

scanf("%d",&ch);

switch(ch){

case 1:

entry();

break;

case 2:

report();

break;

case 3:

masters();

break;

case 4:

exit(1);

break;

default:

printf("Enter valid choice....\n");

}

}

}

void entry(){

while(1){

int ch;

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

printf("\tEntry\n");

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

printf("\t1.Final inspection\n");

printf("\t2.Sale- Billing\n");

printf("\t3. Damage scrap\n");

printf("\t4. back\n");

printf("\t5. exit\n");

printf("\tenter choice");

scanf("%d",&ch);

switch(ch){

case 1:

final\_inspection();

break;

case 2:

sale\_billing();

break;

case 3:

damage();

break;

case 4:

main();

break;

case 5:

exit(1);

break;

default:

printf("\tenter valid choice\n");

}

}

}

void masters(){

while(1){

int ch;

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

printf("\tMasters\n");

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

printf("\t 1.Item Master\n");

printf("\t 2.Party master\n");

printf("\t 3.Back to main menu\n");

printf("\t 4.Exit\n");

printf("\t Enter choice:");

scanf("%d",&ch);

switch(ch){

case 1:

item\_master();

break;

case 2:

party\_master();

break;

case 3:

main();

break;

case 4:

exit(1);

break;

default:

printf("\tenter valid choice!\n");

}

}

}

void item\_master(){

int ch;

while(1){

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

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

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

printf("\t\t1.add item\n");

printf("\t\t2.modify item\n");

printf("\t\t3. delete item\n");

printf("\t\t4.back\n");

printf("\t\t5. exit\n");

printf("\t\t6. print\n");

printf("\t\tenter choice:");

scanf("%d",&ch);

switch(ch){

case 1:

add();

break;

case 2:

modify();

break;

case 3:

del();

break;

case 4:

return;

case 5:

exit(1);

case 6:

print();

break;

default:

printf("\t\tenter valid choice\n");

}

}

}

void party\_master(){

int ch;

while(1){

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

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

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

printf("\t\t1.add item\n");

printf("\t\t2.modify item\n");

printf("\t\tt3. delete item\n");

printf("\t\t4.back\n");

printf("\t\t5. exit\n");

printf("\t\t6. print\n");

printf("\t\tenter choice:");

scanf("%d",&ch);

switch(ch){

case 1:

p\_add();

break;

case 2:

p\_modify();

break;

case 3:

p\_del();

break;

case 4:

return;

case 5:

exit(1);

case 6:

m\_print();

break;

default:

printf("\t\tenter valid choice\n");

}

}

}

void writeReportLinkedList(char filename[], struct report\* head){

FILE\* file;

file = fopen (filename, "ab");

if (file == NULL)

{

fprintf(stderr, "\nCouldn't Open File...'\n");

exit (1);

}

fwrite(head, sizeof(struct report), 1, file);

if(fwrite == 0)

{

printf("Error While Writing\n");

}

fclose(file);

}

struct report\* readReportLinkedList( char filename[]){

struct report\* temp = (struct report \*)malloc(sizeof(struct report));;

struct report\* head; // points to the first node of the linked list in the file

struct report\* last; // points to the last node of the linked list in the file

last = head = NULL;

FILE\* file;

file = fopen (filename, "r");

if (file == NULL)

{

fprintf(stderr, "\nCouldn't Open File11111'\n");

return 0;

}

// reading nodes from the file

// nodes are read in the same order as they were stored

// we are using the data stored in the file to create a new linked list

while(fread(temp, sizeof(struct report), 1, file))

{

if(head==NULL)

{

head = last = (struct report \*)malloc(sizeof(struct report));

}

else

{

last->loc = (struct report \*)malloc(sizeof(struct report));

last = last->loc;

}

last->itemcode = temp->itemcode;

strcpy(last->itemname, temp->itemname);

last->stock=temp->stock;

last->loc = NULL;

}

fclose(file);

return head;

}

void reportadd(struct report\* temp){

struct report \*s1;

if(srep==NULL){

srep=temp;

writeReportLinkedList("Report.txt",srep);

}

else{

s1=srep;

while(s1->loc!=NULL){

s1=s1->loc;

}

s1->loc=temp;

writeReportLinkedList("Report.txt",temp);

}

}

void reportprint(){

struct report \*temp;

temp=readReportLinkedList("Report.txt");

while(temp!=NULL){

printf("\t\t\t\tItem Code:%d\t",temp->itemcode);

printf("\tItem Name:%s\t",temp->itemname);

printf("\n\t\t\t\tStock:%d\t\n",temp->stock);

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

temp=temp->loc;

}

remove("Report.txt");

}

void report(){

time\_t t;

time(&t);

int stock,fi,sale,damage;

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

printf("\t\t\t\t--------Stock Status--------\n");

printf("\t\t\t\tAs On Date:%s\n",ctime(&t));

struct node \*ptr;

ptr=readLinkedList("item.txt");

while(ptr!=NULL){

struct report \*temp;

temp=(struct report\*)malloc(sizeof(struct report));

stock=0;

temp->itemcode=ptr->item\_code;

strcpy(temp->itemname,ptr->item\_name);

fi=fitemsearch(ptr->item\_code);

damage=ditemsearch(ptr->item\_code);

sale=saleitemsearch(ptr->item\_code);

stock=ptr->op\_qty+fi-sale-damage; //stock formula

temp->stock=stock;

temp->loc=NULL;

reportadd(temp);

ptr=ptr->loc;

}

reportprint();

}

**Item Master.h**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node{

int item\_code;

char item\_name[100];

int op\_qty;

int min\_level;

struct node \*loc;

};

struct node \*start=NULL;

void writeLinkedList(char filename[], node\* head){

FILE\* file;

file = fopen (filename, "ab");

if (file == NULL)

{

fprintf(stderr, "\nCouldn't Open File'\n");

exit (1);

}

fwrite(head, sizeof(struct node), 1, file);

if(fwrite != 0)

{

printf("Linked List stored in the file successfully\n");

}

else

{

printf("Error While Writing\n");

}

fclose(file);

}

struct node\* readLinkedList( char filename[]){

struct node\* temp = (struct node \*)malloc(sizeof(struct node));;

struct node\* head; // points to the first node of the linked list in the file

struct node\* last; // points to the last node of the linked list in the file

last = head = NULL;

FILE\* file;

file = fopen (filename, "r");

if (file == NULL)

{

fprintf(stderr, "\nCouldn't Open File...'\n");

return 0;

}

// reading nodes from the file

// nodes are read in the same order as they were stored

// we are using the data stored in the file to create a new linked list

while(fread(temp, sizeof(struct node), 1, file))

{

if(head==NULL)

{

head = last = (struct node \*)malloc(sizeof(struct node));

}

else

{

last->loc = (struct node \*)malloc(sizeof(struct node));

last = last->loc;

}

last->item\_code = temp->item\_code;

strcpy(last->item\_name, temp->item\_name);

last->min\_level=temp->min\_level;

last->op\_qty=temp->op\_qty;

last->loc = NULL;

}

fclose(file);

return head;

}

const char\* isearch(int key){

struct node \*ptr;

int flag=0;

ptr=readLinkedList("item.txt");

while(ptr!=NULL){

if(ptr->item\_code==key){

flag=1;

break;

}

ptr=ptr->loc;

}

if(flag==1){

return(ptr->item\_name);

}

else{

return("0");

}

}

void add(){

const char \*a;

struct node \*ptr,\*temp;

ptr=(struct node \*)malloc(sizeof(struct node));

printf("\t\t\tItem code:");

scanf("%d",&ptr->item\_code);

a=isearch(ptr->item\_code);

if(a=="0"){ // for unique item entry in item master

printf("\t\t\tItem name:");

scanf("%s",&ptr->item\_name);

printf("\t\t\top qty:");

scanf("%d",&ptr->op\_qty);

printf("\t\t\tmin level:");

scanf("%d",&ptr->min\_level);

ptr->loc=NULL;

if(start==NULL){

start=ptr;

writeLinkedList("item.txt",start);

}

else{

temp=start;

while(temp->loc!=NULL){

temp=temp->loc;

}

temp->loc=ptr;

writeLinkedList("item.txt",ptr);

}

}

else{

printf("\t\t\titem already exists!\n");

printf("\t\t\tYou can modify it if you want\n");

}

}

void removeFileContents(const char \*filename,struct node\* newnode) {

FILE \*file = fopen(filename, "w");

if (file == NULL) {

fprintf(stderr, "Couldn't open file: %s\n", filename);

exit(1);

}

while(newnode!=NULL)

{

fwrite(newnode, sizeof(struct node), 1, file);

newnode = newnode->loc;

}

if(fwrite == 0)

{

printf("Error While Writing\n");

}

fclose(file);

}

void modify(){

int n1;

printf("\t\t\tenter item code:");

scanf("%d",&n1);

struct node \*ptr;

struct node \*newnode=readLinkedList("item.txt");

ptr=newnode;

if(newnode==NULL){

printf("\t\t\tNo item to be modified...");

}

else{

while(ptr->item\_code!=n1&&ptr!=NULL){

ptr=ptr->loc;

}

if(ptr->item\_code==n1){

printf("\t\t\tEnter new values:");

printf("\t\t\titem name:");

scanf("%s",&ptr->item\_name);

printf("\t\t\topening qty:");

scanf("%d",&ptr->op\_qty);

printf("\t\t\tminimum level:");

scanf("%d",&ptr->min\_level);

removeFileContents("item.txt",newnode);

}

else

{

printf("\t\t\tnot found the item code...\n");

}

}

}

void del(){

int n1,flag=0;

printf("\t\t\tenter item code to be deleted:");

scanf("%d",&n1);

struct node \*t1=NULL;

struct node \*newnode=readLinkedList("item.txt");

struct node \*ptr;

ptr=newnode;

if(newnode==NULL)

{

printf("\t\t\tNo item to be deleted...\n");

}

else if(ptr->item\_code==n1){

newnode=ptr->loc;

ptr->loc=NULL;

free(ptr);

flag=1;

printf("\t\t\tdeleted successfully...\n");

removeFileContents("item.txt",newnode);

}

else{

t1=ptr;

ptr=ptr->loc;

while(ptr!=NULL)

{

if(ptr->item\_code==n1)

{

t1->loc=ptr->loc;

ptr->loc=NULL;

free(ptr); //delete the node

printf("\t\t\tdeleted successfully..\n");

flag=1;

break;

}

t1=ptr;

ptr=ptr->loc;

}

struct node \*rem=newnode;

while(rem!=NULL){

printf("%s\n",rem->item\_name);

rem=rem->loc;

}

removeFileContents("item.txt",newnode);

}

if(flag==0){

printf("\t\t\titem code doesnt exist..\n");

}

}

void print(){

struct node \*newnode=readLinkedList("item.txt");

struct node \*ptr;

ptr=newnode;

while(ptr!=NULL){

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

printf("\t\t\tItem code:%d\n",ptr->item\_code);

printf("\t\t\tItem name:%s\n",ptr->item\_name);

printf("\t\t\tMinimum Level:%d\n",ptr->min\_level);

printf("\t\t\tOpening Qty:%d\n",ptr->op\_qty);

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

ptr=ptr->loc;

}

}

**Party Master.h**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct pnode{

int GSTNo;

char party\_name[100];

int mobile;

char address[100];

char email[100];

struct pnode \*loc;

};

struct pnode \*pstart=NULL;

void writepartyLinkedList(char filename[], pnode\* head){

FILE\* file;

file = fopen (filename, "ab");

if (file == NULL)

{

fprintf(stderr, "\nCouldn't Open File'\n");

exit (1);

}

fwrite(head, sizeof(struct pnode), 1, file);

if(fwrite == 0)

{

printf("Error While Writing\n");

}

fclose(file);

}

struct pnode\* readpartyLinkedList( char filename[]){

struct pnode\* temp = (struct pnode \*)malloc(sizeof(struct pnode));;

struct pnode\* head; // points to the first node of the linked list in the file

struct pnode\* last; // points to the last node of the linked list in the file

last = head = NULL;

FILE\* file;

file = fopen (filename, "r");

if (file == NULL)

{

fprintf(stderr, "\nCouldn't Open File'\n");

return 0;

}

// reading nodes from the file

// nodes are read in the same order as they were stored

// we are using the data stored in the file to create a new linked list

while(fread(temp, sizeof(struct pnode), 1, file))

{

if(head==NULL)

{

head = last = (struct pnode \*)malloc(sizeof(struct pnode));

}

else

{

last->loc = (struct pnode \*)malloc(sizeof(struct pnode));

last = last->loc;

}

last->GSTNo = temp->GSTNo;

strcpy(last->party\_name, temp->party\_name);

last->mobile=temp->mobile;

strcpy(last->address, temp->address);

strcpy(last->email, temp->email);

last->loc = NULL;

}

fclose(file);

return head;

}

void removePartyFileContents(const char \*filename,struct pnode\* newnode) {

FILE \*file = fopen(filename, "w");

if (file == NULL) {

fprintf(stderr, "Couldn't open file: %s\n", filename);

exit(1);

}

while(newnode!=NULL)

{

fwrite(newnode, sizeof(struct pnode), 1, file);

newnode = newnode->loc;

}

if(fwrite == 0)

{

printf("Error While Writing\n");

}

fclose(file);

}

int psearch(const char \*key1){

struct pnode \*ptr;

int flag=0;

ptr=readpartyLinkedList("party.txt");

while(ptr!=NULL){

if(strcmp(key1,ptr->party\_name)==0){

flag=1;

break;

}

ptr=ptr->loc;

}

return flag;

}

int uniquegst(int gst){

int flag=0;

struct pnode \*t=readpartyLinkedList("party.txt");

while(t!=NULL){

if(t->GSTNo==gst){

flag=1;

break;

}

t=t->loc;

}

return flag;

}

void p\_add(){

int a;

struct pnode \*ptr,\*temp;

ptr=(struct pnode \*)malloc(sizeof(struct pnode));

printf("\t\t\tGstNo.:");

scanf("%d",&ptr->GSTNo);

a=uniquegst(ptr->GSTNo);

printf("%d",a);

if(a==0){

printf("\t\t\tparty name:");

scanf("%s",&ptr->party\_name);

printf("\t\t\tmobile:");

scanf("%d",&ptr->mobile);

printf("\t\t\taddress:");

scanf("%s",&ptr->address);

printf("\t\t\t E-mail:");

scanf("%s",&ptr->email);

ptr->loc=NULL;

if(pstart==NULL){

pstart=ptr;

writepartyLinkedList("party.txt",pstart);

}

else{

temp=pstart;

while(temp->loc!=NULL){

temp=temp->loc;

}

temp->loc=ptr;

writepartyLinkedList("party.txt",ptr);

}

}

else{

printf("\t\t\tThere exist a party with this Gst no.\n");

printf("\t\t\tYou can modify it if you want\n");

}

}

void p\_modify(){

int n1;

printf("\t\t\tenter Gst No:");

scanf("%d",&n1);

struct pnode \*ptr;

struct pnode \*newnode=readpartyLinkedList("party.txt");

ptr=newnode;

if(newnode==NULL){

printf("\t\t\tNo item to be modified...\n");

}

else{

while(ptr->GSTNo!=n1&&ptr!=NULL)

{

ptr=ptr->loc;

}

if(ptr->GSTNo==n1)

{

printf("\t\t\tEnter new values:\n");

printf("\t\t\tparty name:");

scanf("%s",&ptr->party\_name);

printf("\t\t\tmobile:");

scanf("%d",&ptr->mobile);

printf("\t\t\taddress:");

scanf("%s",&ptr->address);

printf("\t\t\tE-mail");

scanf("%s",ptr->email);

removePartyFileContents("party.txt",newnode);

}

else{

printf("\t\t\tnot found the item code.\n");

}

}

}

void p\_del(){

int n1,flag=0;

printf("\t\t\tenter Gst NO. to be deleted:");

scanf("%d",&n1);

struct pnode \*ptr,\*t1=NULL,\*newnode;

newnode=readpartyLinkedList("party.txt");

ptr=newnode;

if(newnode==NULL)

{

printf("\t\t\tNothing to be deleted...\n");

}

else if(ptr->GSTNo==n1){

newnode=ptr->loc;

ptr->loc=NULL;

free(ptr);

flag=1;

removePartyFileContents("party.txt",newnode);

printf("\t\t\tdeleted successfully...\n");

}

else{

t1=ptr;

ptr=ptr->loc;

while(ptr!=NULL)

{

if(ptr->GSTNo==n1)

{

t1->loc=ptr->loc;

ptr->loc=NULL;

free(ptr); //delete the node

printf("\t\t\tdeleted successfully..\n");

flag=1;

break;

}

t1=ptr;

ptr=ptr->loc;

}

removePartyFileContents("party.txt",newnode);

}

if(flag==0){

printf("\t\t\tParty doesn't exist.\n");

}

}

void m\_print(){

struct pnode \*ptr;

struct pnode \*newnode=readpartyLinkedList("party.txt");

ptr=newnode;

while(ptr!=NULL){

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

printf("\t\t\tGSTNo:%d\n",ptr->GSTNo);

printf("\t\t\tParty:%s\n",ptr->party\_name);

printf("\t\t\tAddress:%s\n",ptr->address);

printf("\t\t\tE-mail%:%s\n",ptr->email);

printf("\t\t\tMobile:%d\n",ptr->mobile);

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

ptr=ptr->loc;

}

}

**Entry.h**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include "item\_masters.h"

#include "party\_masters.h"

void fiprint();

void fiadd();

void fimod();

int fitemsearch(int);

void writeFinalInspLinkedList(char filename[], struct finode\*);

struct finode\* readFinalInspLinkedList(char filename[]);

void damageprint();

void damageadd();

void damagemod();

int ditemsearch(int);

void writeDamageLinkedList(char filename[], struct dnode\*);

struct dnode\* readDamageLinkedList(char filename[]);

void saleadd();

void saleprint();

int saleitemsearch(int);

void tableadd(struct bill \*);

struct finode{

int inspNo;

int SNo;

int itemcode;

char itemname[100];

int qty;

char date[20];

struct finode \* loc;

};

struct dnode{

int DamageNo;

int SNo;

int itemcode;

char itemname[100];

int qty;

char date[20];

struct dnode \* loc;

};

struct snode{

int inv\_no;

char invdate[20];

char party[100];

int total;

int gst\_per;

int gst\_val;

int net;

struct snode \* loc;

};

struct bill{

int inv\_no;

int sno;

int itemcode;

char itemname[100];

int qty;

int rate;

int amt;

struct bill \* link;

};

struct dnode \*dstart=NULL;

struct finode \*fistart=NULL;

struct snode \*sstart=NULL;

struct bill \*sbill=NULL;

final\_inspection()

{ int ch;

while(1){

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

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

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

printf("\t\t\t1.Add\n");

printf("\t\t\t2. Modify\n");

printf("\t\t\t3. Back\n");

printf("\t\t\t4. exit\n");

printf("\t\t\t5. print\n");

printf("\t\t\tEnter choice");

scanf("%d",&ch);

switch(ch){

case 1:

fiadd();

break;

case 2:

fimod();

break;

case 3:

return 0;

break;

case 4:

exit(1);

break;

case 5:

fiprint();

break;

default:

printf("\t\t\tenter valid operation\n");

}

}

}

void removeFinalInspFileContents(const char \*filename,struct finode\* newnode) {

FILE \*file = fopen(filename, "w");

if (file == NULL) {

fprintf(stderr, "Couldn't open file: %s\n", filename);

exit(1);

}

while(newnode!=NULL)

{

fwrite(newnode, sizeof(struct finode), 1, file);

newnode = newnode->loc;

}

if(fwrite == 0)

{

printf("Error While Writing\n");

}

fclose(file);

}

void fiadd(){

struct finode \*ptr,\*temp;

const char \*a;

ptr=(struct finode \*)malloc(sizeof(struct finode));

printf("\t\t\t\tInspection No.:");

scanf("%d",&ptr->inspNo);

printf("\t\t\t\tItem code:");

scanf("%d",&ptr->itemcode);

printf("\t\t\t\tS.No");

scanf("%d",&ptr->SNo);

a=isearch(ptr->itemcode); //search the item code from item master

if(a!="0"){

printf("\t\t\t\titem name:%s\n",a); //fetch item name from item master if exist

strcpy(ptr->itemname,a);

printf("\t\t\t\tquantity:");

scanf("%d",&ptr->qty);

printf("\t\t\t\tInspdate:");

scanf("%s",&ptr->date);

ptr->loc=NULL;

if(fistart==NULL){

fistart=ptr;

writeFinalInspLinkedList("finalinspection.txt",fistart);

}

else{

temp=fistart;

while(temp->loc!=NULL){

temp=temp->loc;

}

temp->loc=ptr;

writeFinalInspLinkedList("finalinspection.txt",ptr);

}

}

else{

printf("\t\t\t\titem code doesn't exist in item master\n"); //else return back to main menu

}

}

void fimod(){

int n1;

printf("\t\t\t\tenter Inspection No.:");

scanf("%d",&n1);

struct finode \*newnode=readFinalInspLinkedList("finalinspection.txt");

struct finode \*ptr=newnode;

if(newnode==NULL){

printf("\t\t\t\tNo item to be modified...");

}

else{

while(ptr->inspNo!=n1&&ptr!=NULL){

ptr=ptr->loc;

}

if(ptr->inspNo==n1){

printf("\t\t\t\tEnter new values:\n");

printf("\t\t\t\titem code:");

scanf("%d",&ptr->itemcode);

const char \*a=isearch(ptr->itemcode); //search the item code from item master

if(a!="0"){

printf("\t\t\t\titem name:%s\n",a); //fetch item name from item master if exist

strcpy(ptr->itemname,a);

printf("\t\t\t\tquantity");

scanf("%d",&ptr->qty);

printf("\t\t\t\tinspection date");

scanf("%s",&ptr->date);

removeFinalInspFileContents("finalinspection.txt",newnode);

}

else{

printf("\t\t\t\titem code doesn't exist\n");

}

}

else

{

printf("\t\t\t\tnot found the inspection number..");

}

}

}

int fitemsearch(int key){

int qty=0;

struct finode \*temp=readFinalInspLinkedList("finalinspection.txt");

while(temp!=NULL){

if(temp->itemcode==key){

qty+=temp->qty;

}

temp=temp->loc;

}

return qty;

}

void writeFinalInspLinkedList(char filename[], finode\* head){

FILE\* file;

file = fopen (filename, "ab");

if (file == NULL)

{

fprintf(stderr, "\nCouldn't Open File'\n");

exit (1);

}

fwrite(head, sizeof(struct finode), 1, file);

if(fwrite == 0)

{

printf("Error While Writing\n");

}

fclose(file);

}

struct finode\* readFinalInspLinkedList( char filename[]){

struct finode\* temp = (struct finode \*)malloc(sizeof(struct finode));;

struct finode\* head; // points to the first node of the linked list in the file

struct finode\* last; // points to the last node of the linked list in the file

last = head = NULL;

FILE\* file;

file = fopen (filename, "r");

if (file == NULL)

{

fprintf(stderr, "\nCouldn't Open File'\n");

return 0;

}

// reading nodes from the file

// nodes are read in the same order as they were stored

// we are using the data stored in the file to create a new linked list

while(fread(temp, sizeof(struct finode), 1, file))

{

if(head==NULL)

{

head = last = (struct finode \*)malloc(sizeof(struct finode));

}

else

{

last->loc = (struct finode \*)malloc(sizeof(struct finode));

last = last->loc;

}

last->inspNo = temp->inspNo;

strcpy(last->itemname, temp->itemname);

last->itemcode=temp->itemcode;

strcpy(last->date, temp->date);

last->qty=temp->qty;

last->SNo=temp->SNo;

last->loc = NULL;

}

fclose(file);

return head;

}

void fiprint(){

struct finode \*ptr=readFinalInspLinkedList("finalinspection.txt");

while(ptr!=NULL)

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

printf("\t\t\t\tInspNo:%d\n",ptr->inspNo);

printf("\t\t\t\tItem Code:%d\n",ptr->itemcode);

printf("\t\t\t\tItem Name:%s\n",ptr->itemname);

printf("\t\t\t\tDate:%s\n",ptr->date);

printf("\t\t\t\tQty:%d\n",ptr->qty);

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

ptr=ptr->loc;

}

}

////DAMAGE function

damage(){

int ch;

while(1){

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

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

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

printf("\t\t\t1.Add\n");

printf("\t\t\t2. Modify\n");

printf("\t\t\t3. Back\n");

printf("\t\t\t4. exit\n");

printf("\t\t\t5. print\n");

printf("\t\t\tEnter choice");

scanf("%d",&ch);

switch(ch){

case 1:

damageadd();

break;

case 2:

damagemod();

break;

case 3:

return 0;

break;

case 4:

exit(1);

break;

case 5:

damageprint();

break;

default:

printf("\t\t\tenter valid operation\n");

}

}

}

void damageprint(){

struct dnode \*newnode=readDamageLinkedList("damage.txt");

struct dnode \*ptr=newnode;

while(ptr!=NULL)

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

printf("\t\t\t\tDamageNo:%d\n",ptr->DamageNo);

printf("\t\t\t\tSNo:%d\n",ptr->SNo);

printf("\t\t\t\tItem Code:%d\n",ptr->itemcode);

printf("\t\t\t\tItem Name:%s\n",ptr->itemname);

printf("\t\t\t\\tDate:%s\n",ptr->date);

printf("\t\t\t\tQuantity:%d\n",ptr->qty);

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

ptr=ptr->loc;

}

}

void damageadd(){

struct dnode \*ptr,\*temp;

const char \*a;

ptr=(struct dnode \*)malloc(sizeof(struct dnode));

printf("\t\t\t\tDamage No.");

scanf("%d",&ptr->DamageNo);

printf("\t\t\t\tItem code:");

scanf("%d",&ptr->itemcode);

a=isearch(ptr->itemcode); //search the item code from item master

if(a!="0"){

strcpy(ptr->itemname,a);

printf("\t\t\t\tItem name:%s",a);

printf("\t\t\t\tquantity:");

scanf("%d",&ptr->qty);

printf("\t\t\t\tDamage date:");

scanf("%s",&ptr->date);

printf("\t\t\t\tS.No");

scanf("%d",&ptr->SNo);

ptr->loc=NULL;

if(dstart==NULL){

dstart=ptr;

writeDamageLinkedList("damage.txt",dstart);

}

else{

temp=dstart;

while(temp->loc!=NULL){

temp=temp->loc;

}

temp->loc=ptr;

writeDamageLinkedList("damage.txt",ptr);

}

}

else{

printf("\t\t\t\titem code doesn't exist\n"); //else return back to main menu

}

}

void removeDamageFileContents(const char \*filename,struct dnode\* newnode) {

FILE \*file = fopen(filename, "w");

if (file == NULL) {

fprintf(stderr, "Couldn't open file: %s\n", filename);

exit(1);

}

while(newnode!=NULL)

{

fwrite(newnode, sizeof(struct dnode), 1, file);

newnode = newnode->loc;

}

if(fwrite==0)

{

printf("Error While Writing\n");

}

fclose(file);

}

void damagemod(){

int n1;

printf("\t\t\t\tenter Damage No.:");

scanf("%d",&n1);

struct dnode \*ptr,\*newnode;

newnode=readDamageLinkedList("damage.txt");

ptr=newnode;

if(newnode==NULL){

printf("\t\t\t\tNo item to be modified...");

}

else{

while(ptr->DamageNo!=n1&&ptr!=NULL){

ptr=ptr->loc;

}

if(ptr->DamageNo==n1){

printf("\t\t\t\tEnter new values:\n");

printf("\t\t\t\titem code:");

scanf("%d",&ptr->itemcode);

const char \*a=isearch(ptr->itemcode);

if(a!="0"){

printf("\t\t\t\titem name:%s\n",a); //fetch item name from item master if exist

strcpy(ptr->itemname,a);

printf("\t\t\t\tquantity:");

scanf("%d",&ptr->qty);

printf("\t\t\t\tinspection date:");

scanf("%s",&ptr->date);

removeDamageFileContents("damage.txt",newnode);

}

else{

printf("item code doesn't exist\n");

}

}

else

{

printf("\t\t\t\tnot found the damage number..\n");

}

}

}

int ditemsearch(int key){

int qty=0;

struct dnode \*temp;

temp=readDamageLinkedList("Damage.txt");

while(temp!=NULL){

if(temp->itemcode==key){

qty+=temp->qty;

}

temp=temp->loc;

}

return qty;

}

void writeDamageLinkedList(char filename[], struct dnode\* head){

FILE\* file;

file = fopen (filename, "ab");

if (file == NULL)

{

fprintf(stderr, "\nCouldn't Open File'\n");

exit (1);

}

fwrite(head, sizeof(struct dnode), 1, file);

if(fwrite == 0)

{

printf("Error While Writing\n");

}

fclose(file);

}

struct dnode\* readDamageLinkedList( char filename[]){

struct dnode\* temp = (struct dnode \*)malloc(sizeof(struct dnode));;

struct dnode\* head; // points to the first node of the linked list in the file

struct dnode\* last; // points to the last node of the linked list in the file

last = head = NULL;

FILE\* file;

file = fopen (filename, "r");

if (file == NULL)

{

fprintf(stderr, "\nCouldn't Open File+++'\n");

return 0;

}

// reading nodes from the file

// nodes are read in the same order as they were stored

// we are using the data stored in the file to create a new linked list

while(fread(temp, sizeof(struct dnode), 1, file))

{

if(head==NULL)

{

head = last = (struct dnode \*)malloc(sizeof(struct dnode));

}

else

{

last->loc = (struct dnode \*)malloc(sizeof(struct dnode));

last = last->loc;

}

last->DamageNo = temp->DamageNo;

strcpy(last->itemname, temp->itemname);

last->itemcode=temp->itemcode;

last->qty=temp->qty;

last->SNo=temp->SNo;

strcpy(last->date,temp->date);

last->loc = NULL;

}

fclose(file);

return head;

}

//sale billing function

sale\_billing(){

int ch;

while(1){

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

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

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

printf("\t\t\t1.Add\n");

printf("\t\t\t2. Back\n");

printf("\t\t\t3. exit\n");

printf("\t\t\t4. print\n");

printf("\t\t\tEnter choice");

scanf("%d",&ch);

switch(ch){

case 1:

saleadd();

break;

case 2:

return 0;

break;

case 3:

exit(1);

break;

case 4:

saleprint();

break;

default:

printf("\t\t\tenter valid operation\n");

}

}

}

void writeTableBillLinkedList(char filename[], bill\* head){

FILE\* file;

file = fopen (filename, "ab");

if (file == NULL)

{

fprintf(stderr, "\nCouldn't Open File'\n");

exit (1);

}

fwrite(head,sizeof(struct bill),1,file);

if(fwrite==0)

{

printf("Error While Writing\n");

}

fclose(file);

}

void tableadd(struct bill \*s1){

struct bill \*t;

if(sbill==NULL){

sbill=s1;

writeTableBillLinkedList("bill.txt",sbill); //storing bills in file

}

else{

t=sbill;

while(t->link!=NULL){

t=t->link;

}

t->link=s1;

writeTableBillLinkedList("bill.txt",s1); //storing bills in file

}

}

void writeSalesLinkedList(char filename[], snode\* head){

FILE\* file;

file = fopen (filename, "ab");

if (file == NULL)

{

fprintf(stderr, "\nCouldn't Open File'\n");

exit (1);

}

fwrite(head,sizeof(struct snode),1,file);

if(fwrite == 0)

{

printf("Error While Writing\n");

}

fclose(file);

}

struct snode\* readSalesLinkedList( char filename[]){

struct snode\* temp = (struct snode \*)malloc(sizeof(struct snode));;

struct snode\* head; // points to the first node of the linked list in the file

struct snode\* last; // points to the last node of the linked list in the file

last = head = NULL;

FILE\* file;

file = fopen (filename, "r");

if (file == NULL)

{

fprintf(stderr, "\nCouldn't Open File'\n");

return 0;

}

// reading nodes from the file

// nodes are read in the same order as they were stored

// we are using the data stored in the file to create a new linked list

while(fread(temp, sizeof(struct snode), 1, file))

{

if(head==NULL)

{

head = last = (struct snode \*)malloc(sizeof(struct snode));

}

else

{

last->loc = (struct snode \*)malloc(sizeof(struct snode));

last = last->loc;

}

last->inv\_no = temp->inv\_no;

strcpy(last->invdate, temp->invdate);

last->total=temp->total;

strcpy(last->party, temp->party);

last->gst\_per=temp->gst\_per;

last->gst\_val=temp->gst\_val;

last->net=temp->net;

last->loc = NULL;

}

fclose(file);

return head;

}

void saleadd(){

struct bill \*s2;

struct snode \*ptr,\*temp;

int sum,i,a,key,n;

const char \* b;

ptr=(struct snode \*)malloc(sizeof(struct snode));

printf("\t\t\t\tInvoice No.:");

scanf("%d",&ptr->inv\_no);

printf("\t\t\t\tInvoice date:");

scanf("%s",&ptr->invdate);

printf("\t\t\t\tParty:");

scanf("%s",&ptr->party);

a=psearch(ptr->party); //search the party name from party master

if(a==1)

{ sum=0;

printf("\t\t\t\t\tenter number of items:");

scanf("%d",&n);

for(i=0;i<n;i++){

struct bill \*s1;

s1=(struct bill\*)malloc(sizeof(struct bill));

printf("\t\t\t\t\titem code:");

scanf("%d",&key);

b=isearch(key); //search the item code from item master

if(b!="0")

{ s1->inv\_no=ptr->inv\_no;

s1->sno=i+1;

printf("\t\t\t\t\ts.no:%d",s1->sno);

s1->itemcode=key;

strcpy(s1->itemname,b);

printf("\t\t\t\t\t\nquantity:");

scanf("%d",&s1->qty);

printf("\t\t\t\t\t\nrate:");

scanf("%d",&s1->rate);

s1->amt=s1->qty\*s1->rate;

printf("\t\t\t\t\t\namount:%d",s1->amt);

s1->link=NULL;

tableadd(s1);

sum+=s1->amt;

}

else{

i--; //since that item code doesn't exist in it

}

printf("\n");

}

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

ptr->total=sum;

printf("\t\t\t\t\nTotal:%d",ptr->total);

printf("\t\t\t\t\nGst %:");

scanf("%d",&ptr->gst\_per);

ptr->gst\_val=(ptr->gst\_per\*ptr->total)/100;

printf("\t\t\t\t\nGst val:%d",ptr->gst\_val);

ptr->net=ptr->total+ptr->gst\_val;

printf("\n\t\t\t\tNet amt.:%d",ptr->net);

ptr->loc=NULL;

if(sstart==NULL){

sstart=ptr;

writeSalesLinkedList("Sales.txt",sstart);

}

else{

temp=sstart;

while(temp->loc!=NULL){

temp=temp->loc;

}

temp->loc=ptr;

writeSalesLinkedList("Sales.txt",ptr);

}

}

else{

printf("\t\t\t\tparty doesn't exist in party master\n"); //else return back to main menu

}

}

struct bill\* readBilllinkedList(char filename[],int invoice){

struct bill \*temp=(struct bill\*)malloc(sizeof(struct bill));

struct bill \*head;

struct bill \*last;

head=last=NULL;

FILE \*file;

file=fopen(filename,"r");

if(file==NULL){

fprintf(stderr, "\nCouldn't Open File'\n");

return 0;

}

while(fread(temp,sizeof(struct bill),1,file)==1){

if(temp->inv\_no==invoice){

if(head==NULL){

head=last=(struct bill\*)malloc(sizeof(struct bill));

}

else{

last->link=(struct bill \*)malloc(sizeof(struct bill\*));

last=last->link;

}

last->inv\_no=temp->inv\_no;

last->sno=temp->sno;

last->itemcode=temp->itemcode;

strcpy(last->itemname,temp->itemname);

last->qty=temp->qty;

last->rate=temp->rate;

last->amt=temp->amt;

last->link=NULL;

}

}

fclose(file);

return head;

}

void saleprint(){

struct snode \*ptr;

ptr=readSalesLinkedList("Sales.txt");

struct bill \*temp;

while(ptr!=NULL){

printf("\t\t\tInvoice No.:%d\n",ptr->inv\_no);

printf("\t\t\tInvoice Date:%s\n",ptr->invdate);

printf("\t\t\tParty:%s\n",ptr->party);

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

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

temp=readBilllinkedList("bill.txt",ptr->inv\_no);

while(temp!=NULL)

{

printf("\t\t\t\tS.NO:%d\n",temp->sno);

printf("\t\t\t\tItem code:%d\n",temp->itemcode);

printf("\t\t\t\tItem Name:%s\n",temp->itemname);

printf("\t\t\t\tQty:%d\n",temp->qty);

printf("\t\t\t\tRate:%d\n",temp->rate);

printf("\t\t\t\tAmt:%d\n",temp->amt);

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

temp=temp->link;

}

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

printf("\t\t\t Total:%d\n",ptr->total);

printf("\t\t\t Gst percentage%:%d%\n",ptr->gst\_per);

printf("\t\t\t Gst Amount:%d\n",ptr->gst\_val);

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

printf("\t\t\tNet amount:%d\n",ptr->net);

ptr=ptr->loc;

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

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

}

}

int saleitemsearch(int key){

struct snode \*s1;

s1=readSalesLinkedList("Sales.txt");

struct bill \*t;

int qty=0;

while(s1!=NULL){

t=readBilllinkedList("bill.txt",s1->inv\_no);

while(t!=NULL){

if(t->itemcode==key){

qty+=t->qty;

}

t=t->link;

}

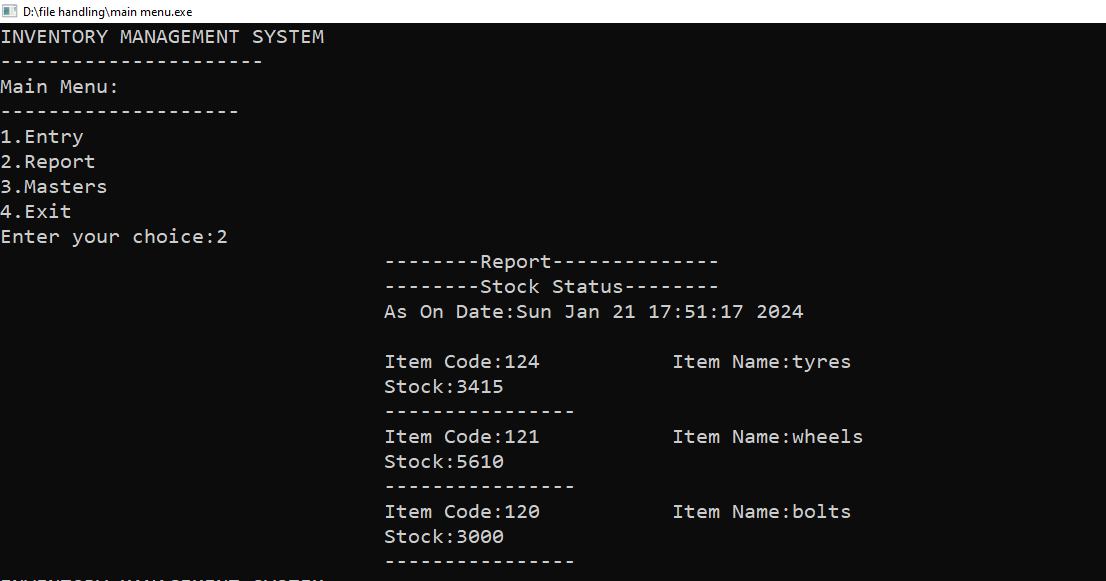
s1=s1->loc;

}

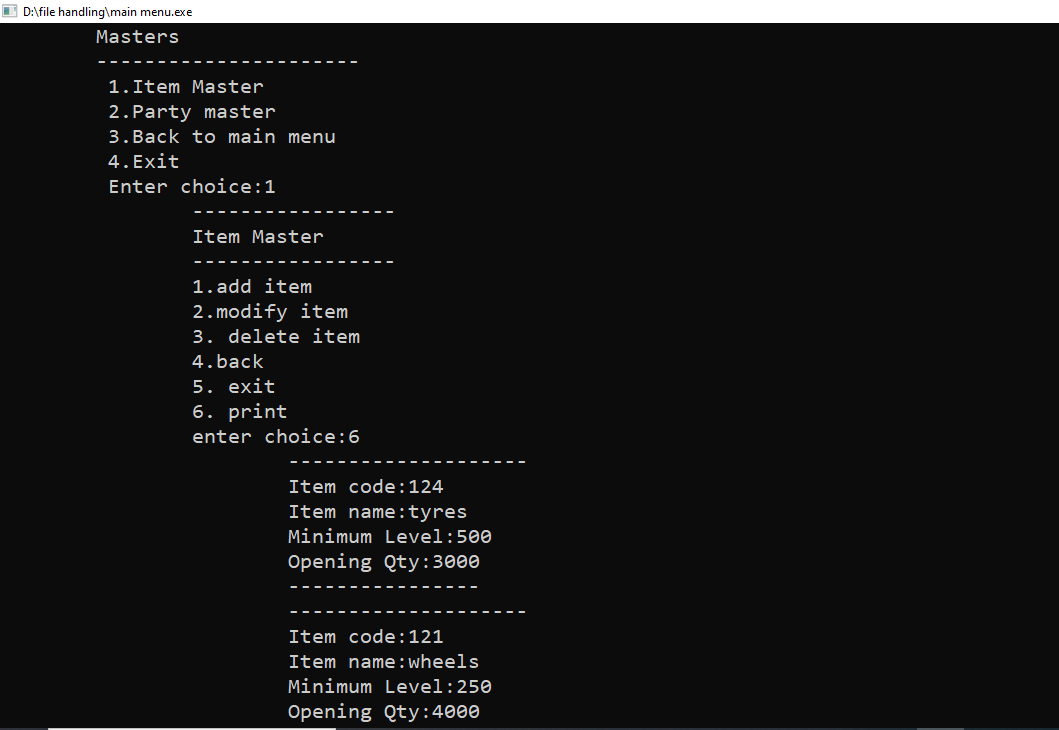
return qty;

}

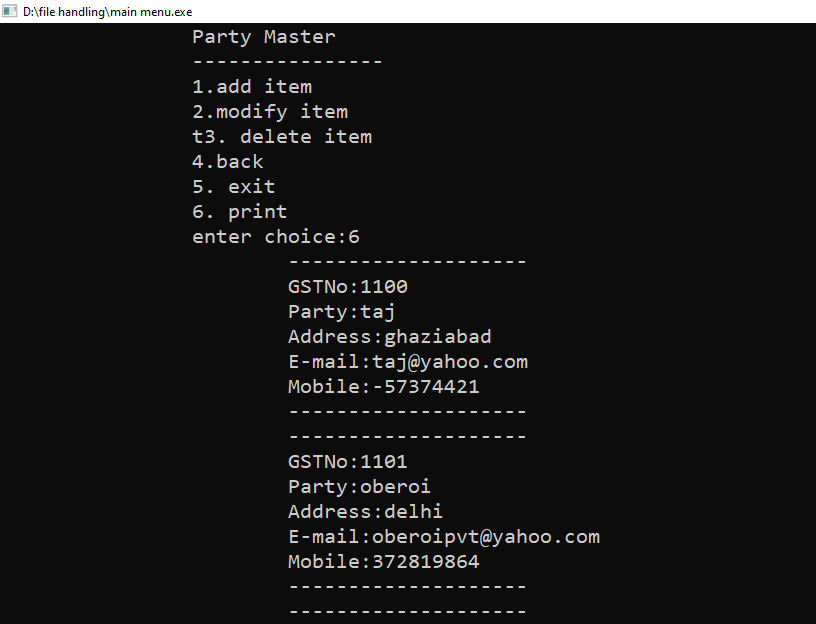
**Output**

***Stock Report as on date***

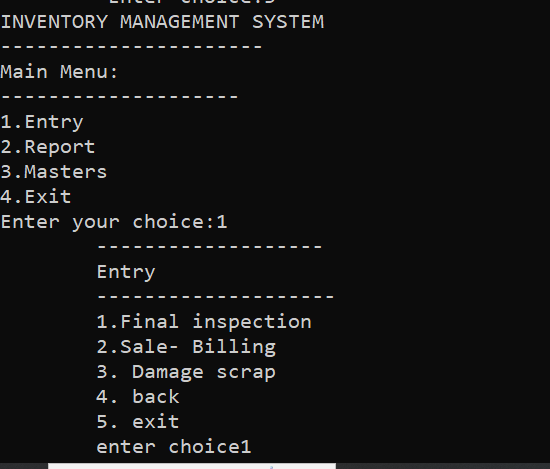
***Item Master***



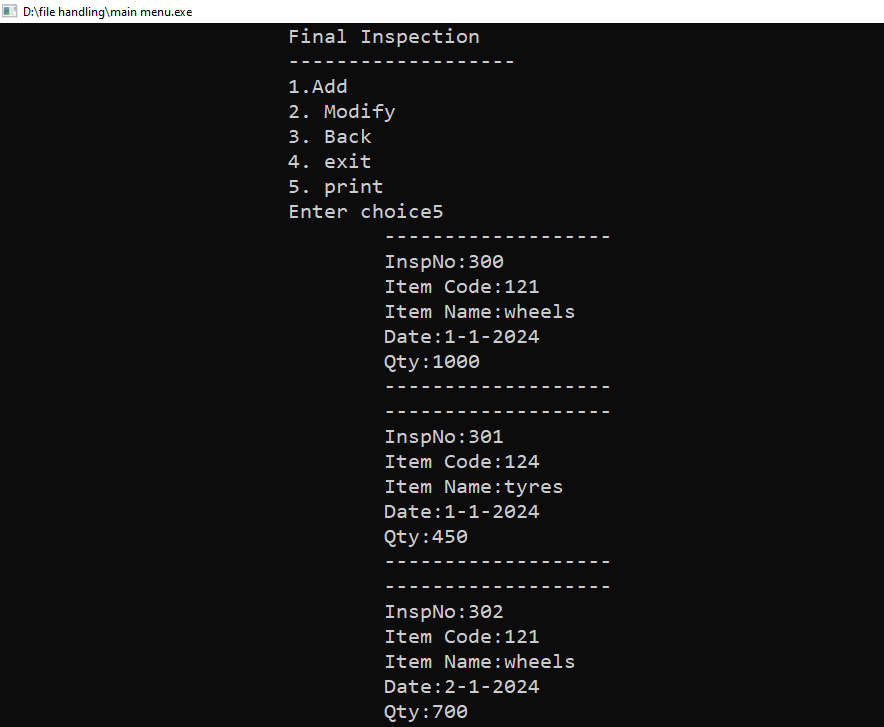
***Party Masters***



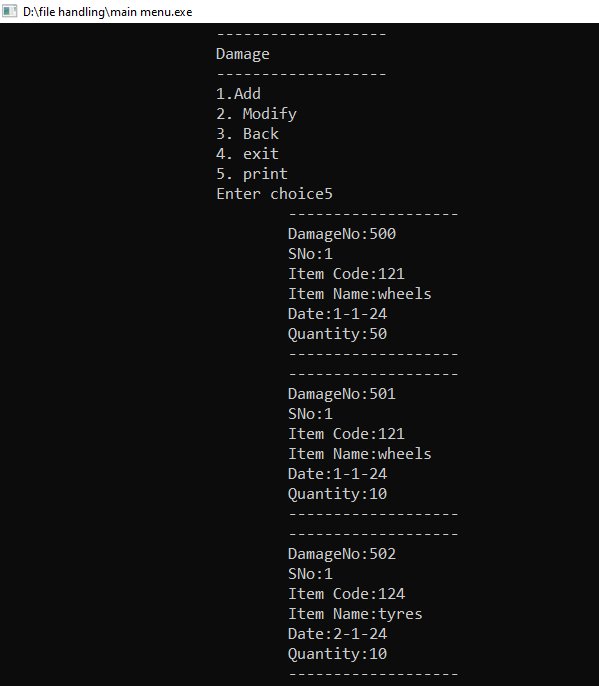
***Entry.h***



***a)Final Inspected Goods***



***b)Damage goods***



***c)Sale(Invoice***

