**SSN COLLEGE OF ENGINEERING**

**(Autonomous)**

Affiliated to Anna University

DEPARTMENT OF CSE

**UCS 1312 Data Structures Lab Laboratory**

**Exercise 14: MINI PROJECT**

**=================================================**

Register No.   : 185001071

    Name              : KARTHIKEYAN K

    Class               : CSE-B (SEMESTER-3)

    Academic Year: 2019-2020 Batch: 2018-2022

**ONLINE DEPARTMENT STORE**

**Aim**: To develop an online departmental store.

**CODE:**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

typedef struct mynode

{

char item\_name[50];

float price;

int stock,quantity;

int mm[2],dd[2],yy[2];

struct mynode \*left;

struct mynode \*right;

}node;

node\* find(char name[],node \*t)

{

if (t==NULL)

return NULL;

if(strcmp(name,t->item\_name)<0)

return find(name,t->left);

else if(strcmp(name,t->item\_name)>0)

return find(name,t->right);

else

return t;

}

node\* insert(char name[],float cost,int stock,int mm[2],int dd[2],int yy[2],node \*t)

{

if(t==NULL)

{

t=(node\*)malloc(sizeof(node));

strcpy(t->item\_name,name);

t->price=cost;

t->stock=stock;

t->mm[0]=mm[0];t->mm[1]=mm[1];

t->dd[0]=dd[0];t->dd[1]=dd[1];

t->yy[0]=yy[0];t->yy[1]=yy[1];

t->left=t->right=NULL;

printf("\nThe Item has been successfully added");

}

else if(strcmp(name,t->item\_name)<0)

t->left=insert(name,cost,stock,mm,dd,yy,t->left);

else if(strcmp(name,t->item\_name)>0)

t->right=insert(name,cost,stock,mm,dd,yy,t->right);

return t;

}

node\* insert1(char name[],float cost,int stock,int quantity,int mm[2],int dd[2],int yy[2],node \*t)

{

if(t==NULL)

{

t=(node\*)malloc(sizeof(node));

strcpy(t->item\_name,name);

t->price=cost;

t->stock=stock;

t->quantity=quantity;

t->mm[0]=mm[0];t->mm[1]=mm[1];

t->dd[0]=dd[0];t->dd[1]=dd[1];

t->yy[0]=yy[0];t->yy[1]=yy[1];

t->left=t->right=NULL;

printf("The Item has been successfully added\n\n");

}

else if(strcmp(name,t->item\_name)<0)

t->left=insert1(name,cost,stock,quantity,mm,dd,yy,t->left);

else if(strcmp(name,t->item\_name)>0)

t->right=insert1(name,cost,stock,quantity,mm,dd,yy,t->right);

return t;

}

void inorder(node \*t)

{

if(t!=NULL){

inorder(t->left);

printf("\n");

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

printf("The price :%f\n",t->price);

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

printf("Date of manufacture :%d / %d / %d\n",t->dd[0],t->mm[0],t->yy[0]);

printf("Date of expiry :%d / %d / %d\n",t->dd[1],t->mm[1],t->yy[1]);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

inorder(t->right);

}

}

void inorder1(node \*t)

{

if(t!=NULL){

inorder1(t->left);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

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

printf("The price :%f\n",t->price);

printf("The quantity :%d\n",t->quantity);

printf("Date of manufacture :%d / %d / %d\n",t->dd[0],t->mm[0],t->yy[0]);

printf("Date of expiry :%d / %d / %d\n",t->dd[1],t->mm[1],t->yy[1]);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

inorder1(t->right);

}

}

float total=0.0;

float calculateBILL(node \*t)

{

if(t!=NULL)

{

calculateBILL(t->left);

printf("%s\t%f\t%d\t%f\n",t->item\_name,t->price,t->quantity,(t->price\*t->quantity));

total+=(t->price\*t->quantity);

calculateBILL(t->right);

}

return total;

}

int main()

{

node \*t=NULL;

node \*t1=NULL;

int choice;

char name[50];

float price;int stock;

int mm[2],dd[2],yy[2];

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

printf(" online department store \n");

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

do{

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

printf("1.add an item\n2.list all items\n3.search for specific item\n4.buy items\n5.Bill\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

printf("\n\nenter the choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:{

printf("enter name :");scanf("%s",name);

printf("enter price :");scanf("%f",&price);

printf("enter the stock :");scanf("%d",&stock);

printf("enter the date of manufacture(dd mm yy) :");scanf("%d %d %d",&dd[0],&mm[0],&yy[0]);

printf("enter the date of expiry(dd mm yy) :");scanf("%d %d %d",&dd[1],&mm[1],&yy[1]);

t=insert(name,price,stock,mm,dd,yy,t);

}break;

case 2:{

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

printf(" The list of items available ");

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

inorder(t);break;

}

case 3:{

char name[50];

printf("\nenter the item to search ");

scanf("%s",name);

printf("\n");

node \*p=find(name,t);

if(p!=NULL){

printf("Item name :%s\n",p->item\_name);

printf("The price :%f\n",p->price);

printf("The stock :%d\n",p->stock);

printf("Date of manufacture :%d / %d / %d\n",p->dd[0],p->mm[0],p->yy[0]);

printf("Date of expiry :%d / %d / %d\n",p->dd[1],p->mm[1],p->yy[1]);}

else

{

printf("The item is not available\n");

}

}break;

case 4:{

int n;char item[50];

int ch;int opt;

printf("enter no of items you want to buy\n");

scanf("%d",&n);

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

{

a: printf("\nenter the item %d you want to buy\n",i+1);

scanf("%s",item);

node \*p=find(item,t);

if(p!=NULL)

{

printf("The item name :%s\n",p->item\_name);

printf("The price :%f\n",p->price);

printf("The stock :%d\n",p->stock);

printf("Date of manufacture :%d / %d / %d\n",t->dd[0],t->mm[0],t->yy[0]);

printf("Date of expiry :%d / %d / %d\n\n",t->dd[1],t->mm[1],t->yy[1]);

printf("The item is available \n\n");

printf("Enter the quantity :");

scanf("%d",&(p->quantity));

if(p->quantity>p->stock)

{

printf("The stock is not available\n");

printf("\nThe stock available for %s is only %d\n",p->item\_name,p->stock);

printf("\nEnter a valid quantity you want to buy :");

scanf("%d",&(p->quantity));

t1=insert1(p->item\_name,p->price,p->stock,p->quantity,p->mm,p->dd,p->yy,t1);

p->stock=p->stock-p->quantity;

printf("\nThe item has been bought\n\n");

}

else

{

t1=insert1(p->item\_name,p->price,p->stock,p->quantity,p->mm,p->dd,p->yy,t1);

p->stock=p->stock-p->quantity;

printf("\nThe item has been bought\n\n");

}

}

else

{

printf("The item is not available\n");

printf("\nDo you want to buy any other item(press 1 for yes)\n");

scanf("%d",&opt);

if(opt==1)

{

goto a;

}

}

}

printf("\nThe list of bought items\n");

inorder1(t1);

}break;

case 5:{

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

printf(" CUSTOMER BILL ");

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

printf("ITEM NAME\tPRICE\tQUANTITY\tTOTAL\n");

float total=calculateBILL(t1);

printf("\n\n");

printf("The Total :%f",total);

}

default:exit(0);

}

}while(choice==1||choice==2||choice==3||choice==4||choice==5);

return 0;

}

**Sri Sivasubramaniya Nadar College of Engineering,**

**Kalavakkam – 603 110**

**(An Autonomous Institution, Affiliated to Anna University, Chennai)**

**Department of CSE**

**UCS 1312 –DATA STRUCTURES LABORATORY**

Batch: 2018-2022

Semester: 03

**ONLINE DEPARTMENTAL STORE**

**Mini Project report**

**Karthikeyan K – 185001071**

**Sai Aashish P - 185001130**

**Santhosh R – 185001135**

**ABSTRACT:**

An online departmental store using basic tree concept(binary search tree) was designed and implemented.

The Program contains **5 menu** (add an item, list available items, search an item, buy item, create bill).At first we append all items into a binary search tree from which we can search for a particular item to display it for customer use to check for the date of manufacture and expiry. This tree contains all the items which are available in the store.

The customer must specify the number of items he/she wants to buy. Only the items which are stored into the tree can be bought by the customer and after the item was available in the store(tree),the customer is asked to specify the quantity of item. Only if the quantity is less than stock ,the item would be bought and stored in another tree which contains the list of bought items. After all the items are bought , the list of bought items are printed and the bill is printed along with the total.

**METHODOLOGY**

The different operations that can be performed using this application are:

1. Add items into the store.
2. List all items available in the store.
3. Search for an item in the store and if found display it.
4. Buy items from the store.
5. Print the bill and the total.

The different modules used in implementing these operations are:

**#main.c**

This file contains the main driver program.

**#BST.c**

This file contains the tree ADT and functions like insert, search etc.

**node\* find(char name[],node \*t)**

This function takes two parameters ,the name and the tree ,and searches for name in tree and returns the node which contains name.

**node\* insert(char name[],float cost,intstock,int mm[2],int dd[2],int yy[2],node \*t)**

This function creates a node and assigns parameter values to the node.

The node is then inserted into the tree t and returned.

**node\* insert1(char name[],float cost,intstock,intquantity,int mm[2],int dd[2],int yy[2],node \*t)**

This function is similar to insert function but only the bought items are inserted into the tree.

**void inorder(node \*t)**

This function takes the tree as parameter and displays the items available in the original tree or displays the list of items available in store.

**void inorder1(node \*t)**

This function is similar to inorder() , but only the bought items are listed.

**float calculateBILL(node \*t)**

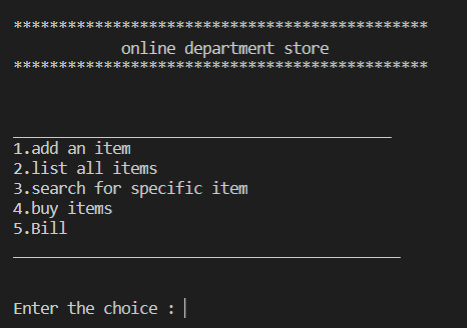
This function takes a tree as parameter and generates a bill from the list of bought items.

**Binary Search Tree**

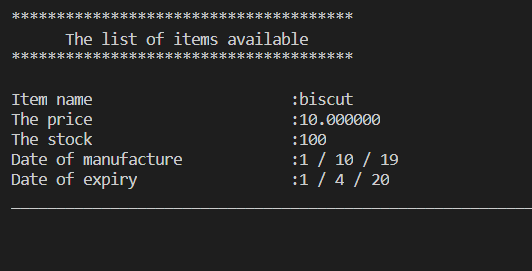
Binary search treeis a node-based binary tree data structure which has the following properties:

* The left subtree of a node contains only nodes with keys lesser than the node’s key.
* The right subtree of a node contains only nodes with keys greater than the node’s key.
* The left and right subtree each must also be a binary search tree.

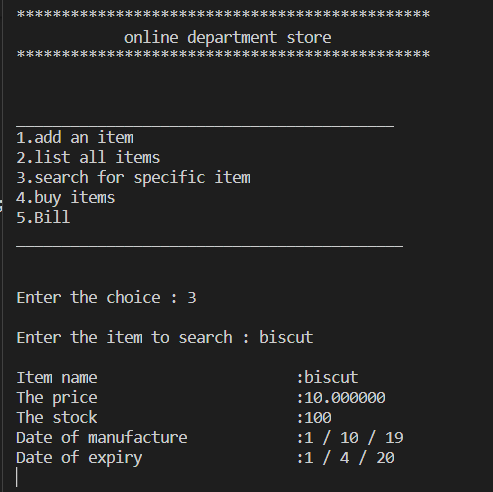
**OUTPUT SCREENSHOTS**



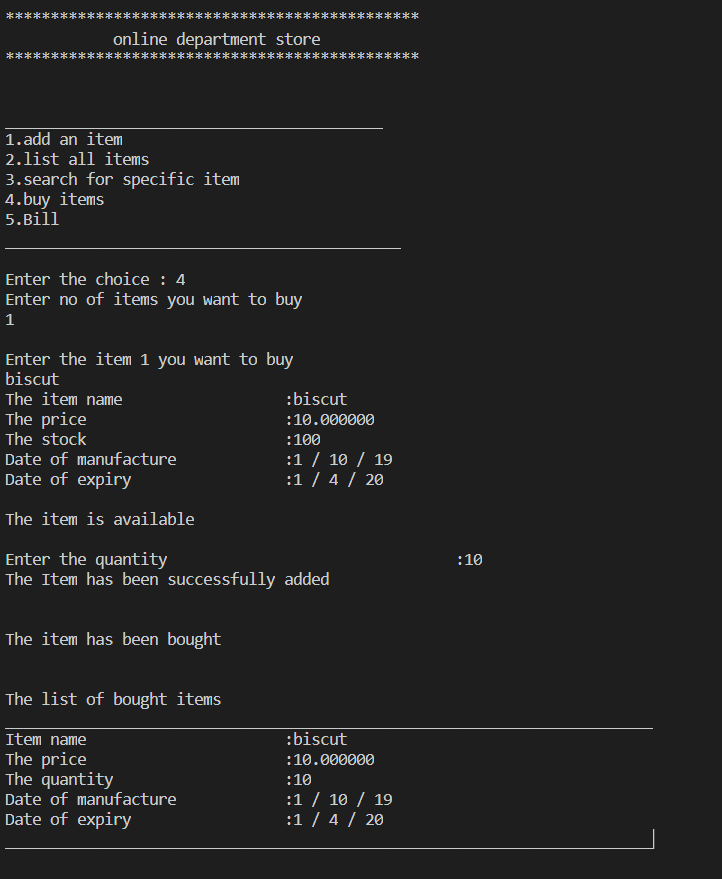
Displaying the options available for the customer.



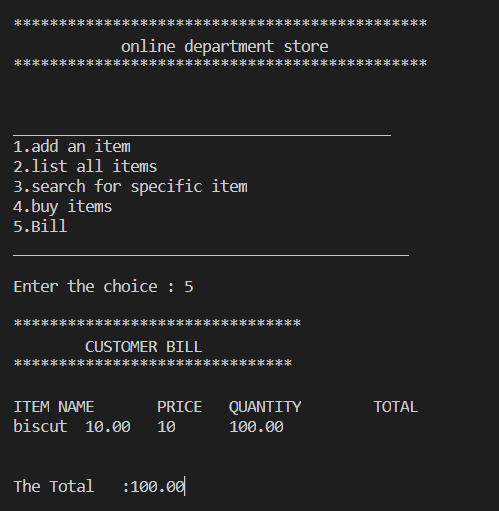
Option 2: Displaying the list of items available in the store.



Option 3:Lets the user to search for an item in store.

****

**Option 4:Lets the user to buy items and after all the items are bought , the list of bought items are displayed.**



Option 5: Generates a bill from the items bought from store.

**CONCLUSION:**

The concept of data structures is implemented in this application and the Online departmental store billing system is efficiently developed.