**UCS 1312 Data Structures Lab**

**DRUGSTORE MANAGEMENT**

Kathiyayini S – 185001072

Kiruthika J – 185001078

Madhumithaa S – 185001087

**Abstract:**

The Drugstore Management application has been built using data structures such as binary search trees and queues. Using this application, the user can perform operations such as add new medicines, update the stock of medicines, search for a particular medicine, display the details of all the medicines available, check for the validity of a particular medicine and sell medicines to a customer.

**Methodology:**

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

1. Add new medicines
2. Update the stock of existing medicines
3. Search for a particular medicine
4. Display all the medicines in the inventory
5. Sell medicines to a customer along with the generation of a bill
6. Check for the validity of a particular medicine

The different modules used in implementing these operations are:

**#main.c** (this file contains the main driver program)

*void disp\_menu()*

The ***disp\_menu()*** function is used to display the menu of all the operations available in the application. **#queue.h** (this file contains the queue ADT and operations implemented using queue)

*int dater(int x)*

The ***dater(int x)***  is a sub-function that is used in preceding functions, used to calculate the number of days between two dates.

*int validate(struct Date m, struct Date e)*

The ***validate(struct Date m, struct Date e)*** function is used to calculate the difference between two dates (which are given as inputs) – today and the expiry date specified for the medicine and hence returns the validity.

*queue\* createQueue(int maxelements)*

The ***createQueue(int maxelements)*** function creates a queue for each drug that is added to the drug inventory, so that new batches can be added, taking the maximum number of elements as input and returning the queue itself.

*int isFull(queue\* q)*

The ***isFull(queue\* q)*** function checks if the specified queue is full and returns 1 if true.

*int isEmpty(queue\* q)*

The ***isEmpty(queue\* q)*** function checks if the specified queue is empty and returns 1 if true.

*void enqueue(queue\* q, int total)*

The ***enqueue(queue\* q, int total)*** functionis used to enqueue a batch of medicines for a particular drug, each time the stock is updated. Hence, it takes the drug’s queue and the amount updated as input.

*void sell(queue \*q, int amt)*

The ***sell(queue \*q, int amt)*** is simply a dequeue function that removes the stock each time a particular drug is purchased, thus reducing the available stock. Hence, it takes the drug’s queue and the amount bought as input.

*void disp(queue \*q)*

The ***disp(queue \*q)*** function displays all the details that have been entered about a particular drug, taking the particular drug’s queue as input.

**#tree.h** (this file contains tree ADT and the operations implemented using tree)

*queue\* give\_queue(tree\* t, char name[])*

The ***give\_queue(tree \*t, char name[])*** function returns the batch of medicine added for a particular drug, retrieving it from the queue. Hence, it takes the drug’s queue and the name of drug as inputs.

*float give\_cost(tree\* t,char name[])*

The ***give\_cost(tree\* t, char name[])*** function returns the cost of a particular drug that has been entered, retrieving it from the binary search tree. Hence, it takes the drug’s queue and the name of drug as inputs.

*tree\* givetree(tree \*t, char name[])*

The ***givetree(tree \*t, char name[])*** function returns the entire tree that consists of details of the particular drug. The details stored in the drug are: ID, name of drug, name of supplier, manufacture date, expiry date, price per tab and number of tabs. Hence, it takes the drug’s queue and the name of drug as inputs.

*void cal\_tot(tree \*t)*

The ***cal\_tot(tree \*t)*** function calculates the total number of medicines.

*tree\* insert(tree \*t,tree \*temp)*

The ***insert(tree \*t, tree \*temp)*** is used to add new drugs to the binary search tree in alphabetical order. Hence, it takes the main binary search tree and a temporary node that contains the drug details as input. The details stored in the drug are: ID, name of drug, name of supplier, manufacture date, expiry date, price per tab and number of tabs.

*void display(tree \*t)*

The ***display(tree \*t)***function is used to display the details in the binary search tree, in this case, all the attributes of the medicine.

*int check(tree \*t, int a)*

The ***check(tree \*t, int a)*** function is used to check the validity of a medicine based on its expiry date and returns true if the medicine has expired.

*int warn(tree \*t)*

The ***warn(tree \*t)*** function is used to warn the user if the medicine has exceeded validity.

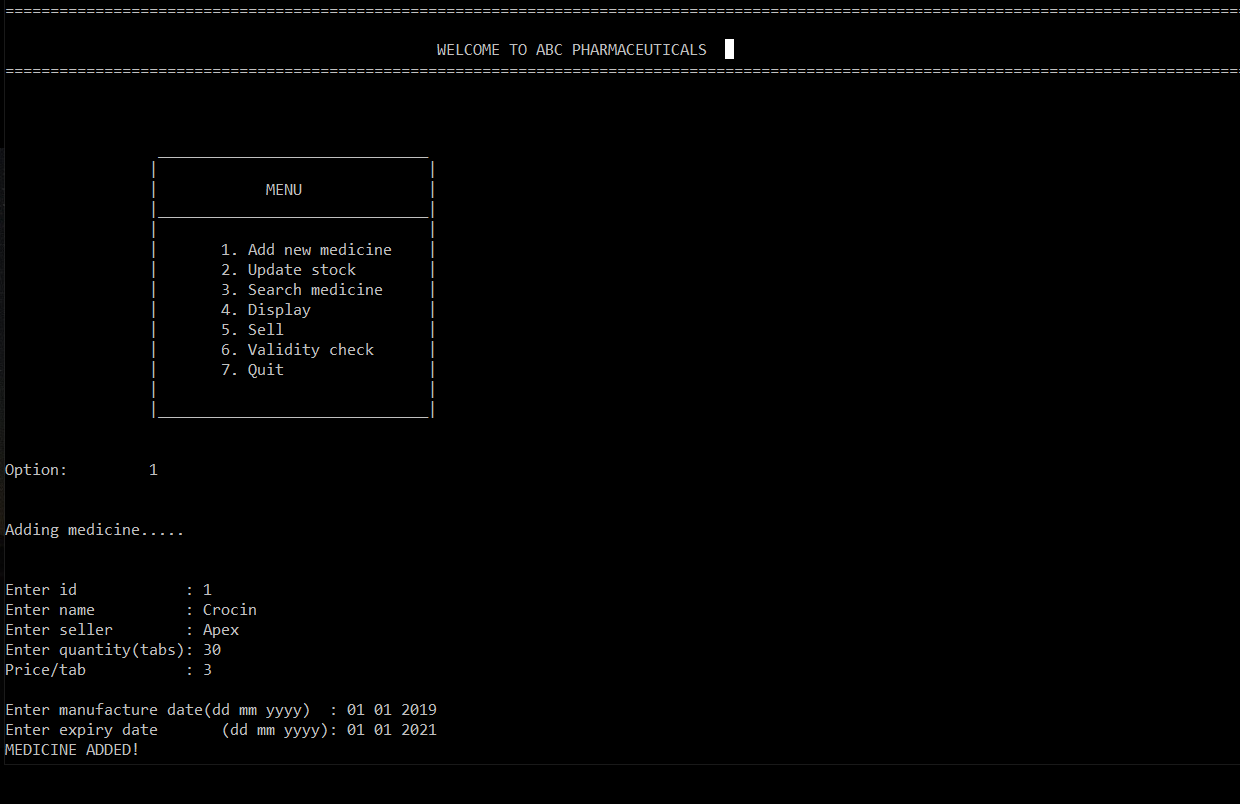
*void search(tree \*t,char name[50])*

The ***search(tree \*t, char name[50])*** function is used to search for a particular drug in the binary search tree and display its details.

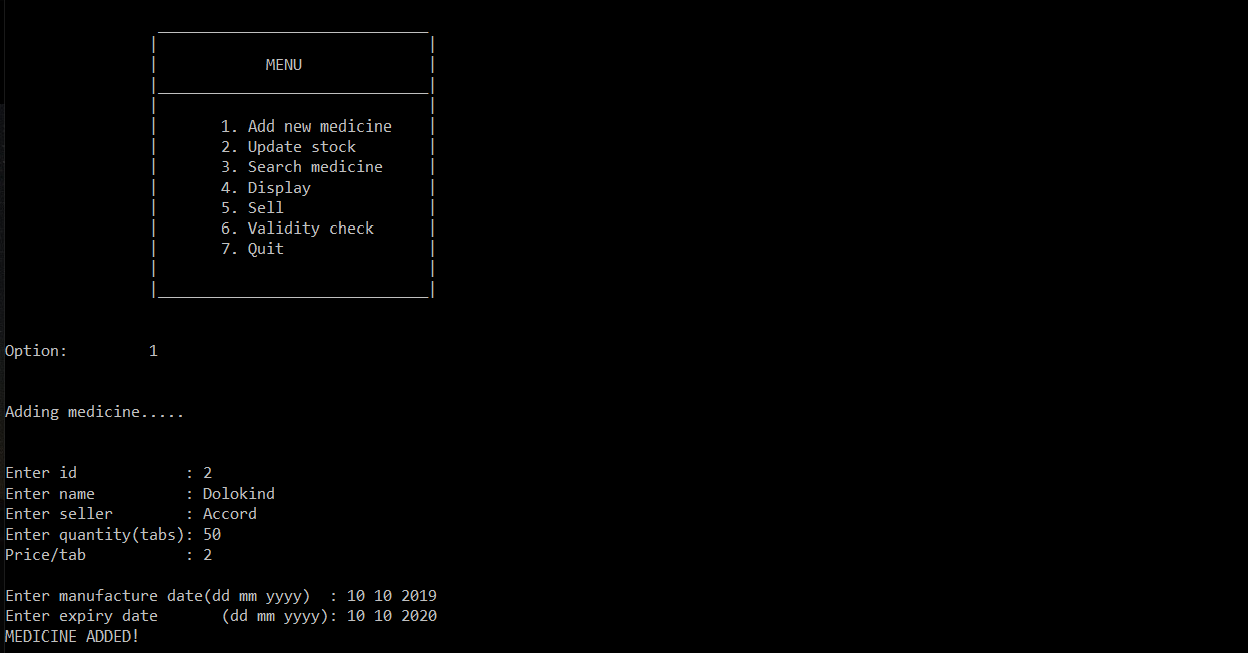
*void inorder(tree \*t)*

The ***inorder(tree \*t)*** function is used to traverse the binary search tree in inorder traversal method.

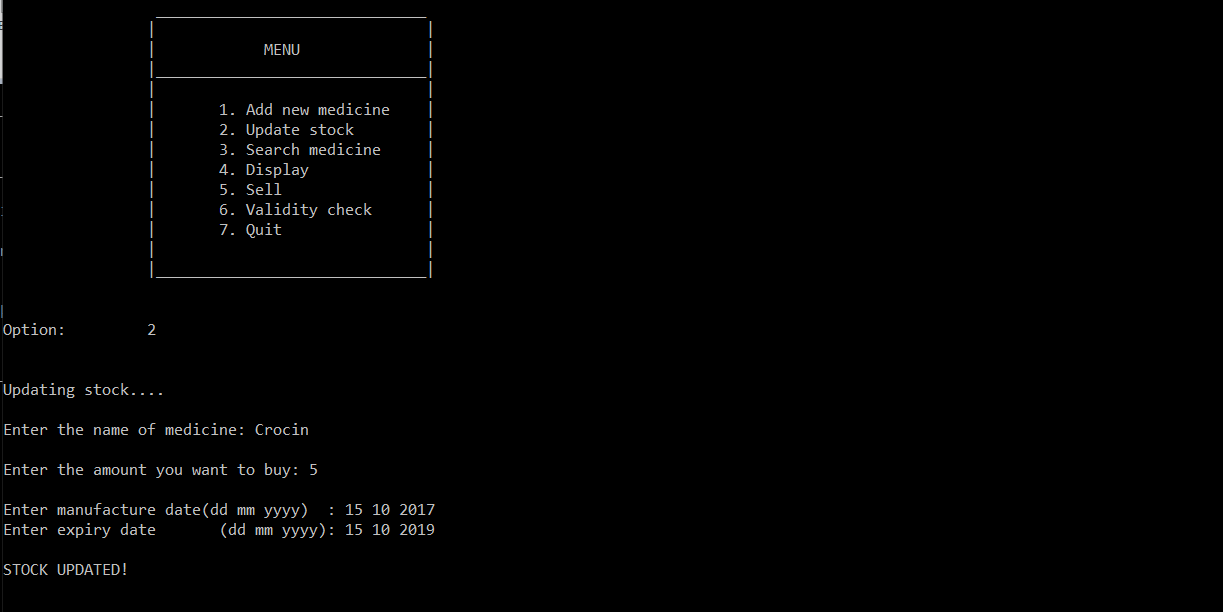
**Output:**



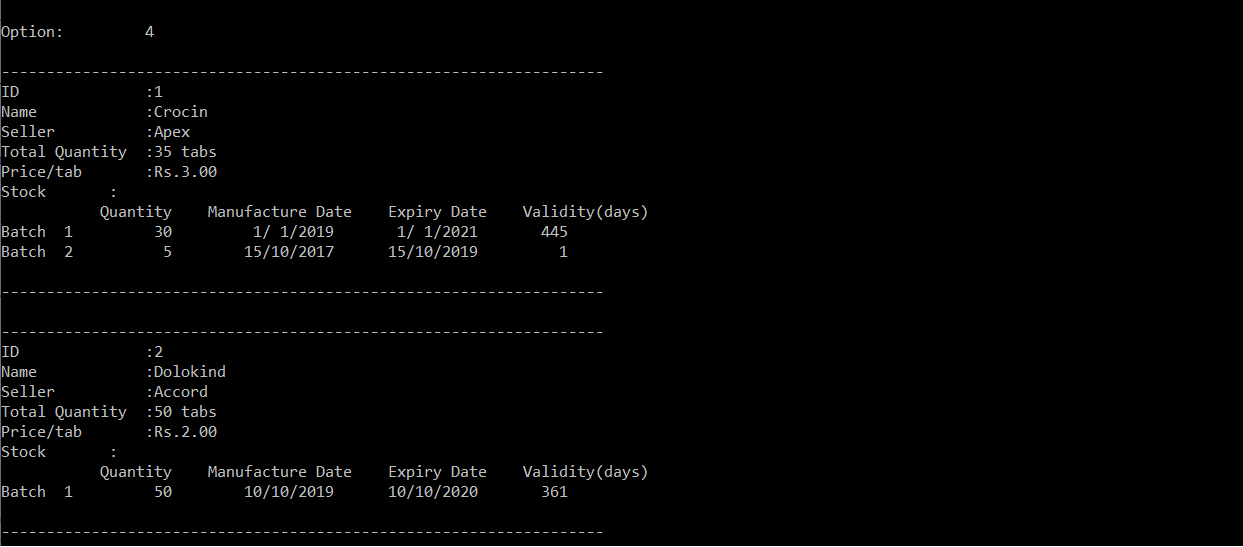
**Option 1: Crocin along with its details are added to the drugstore inventory**



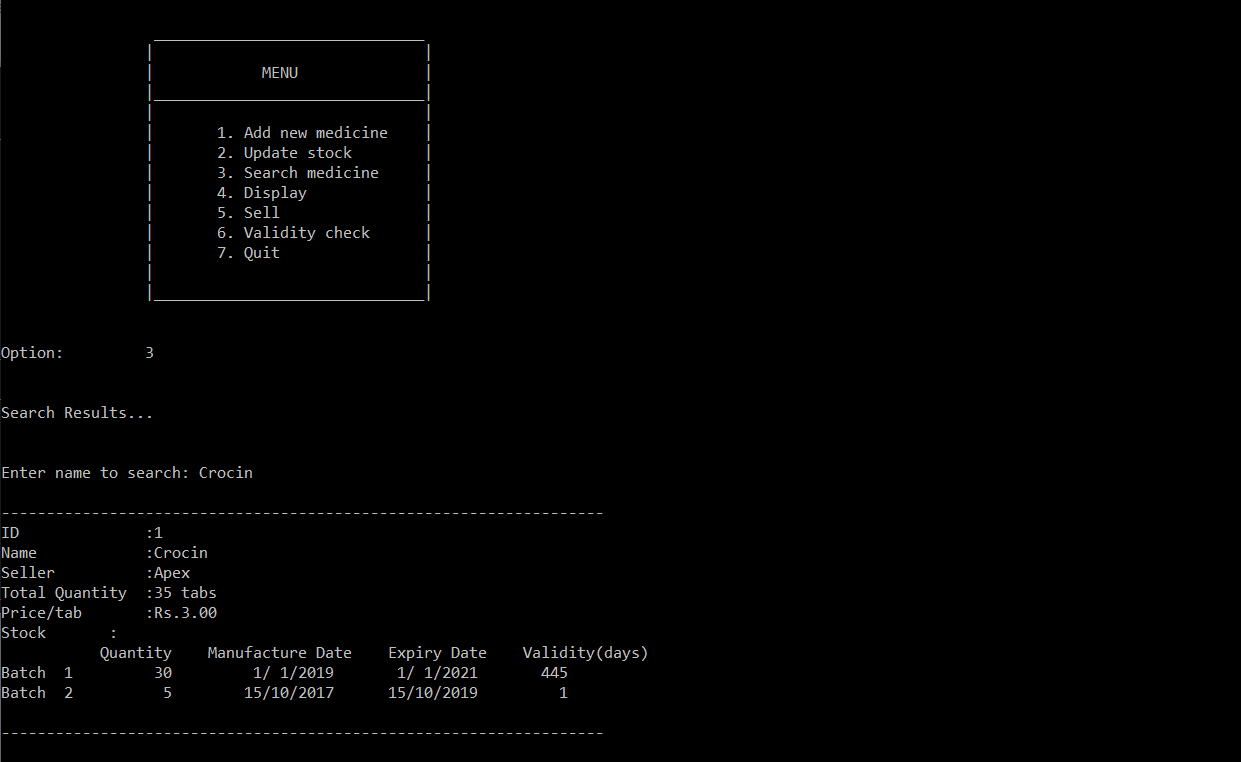
**Option 1: Dolokind along with its attributes are added to the drugstore inventory**



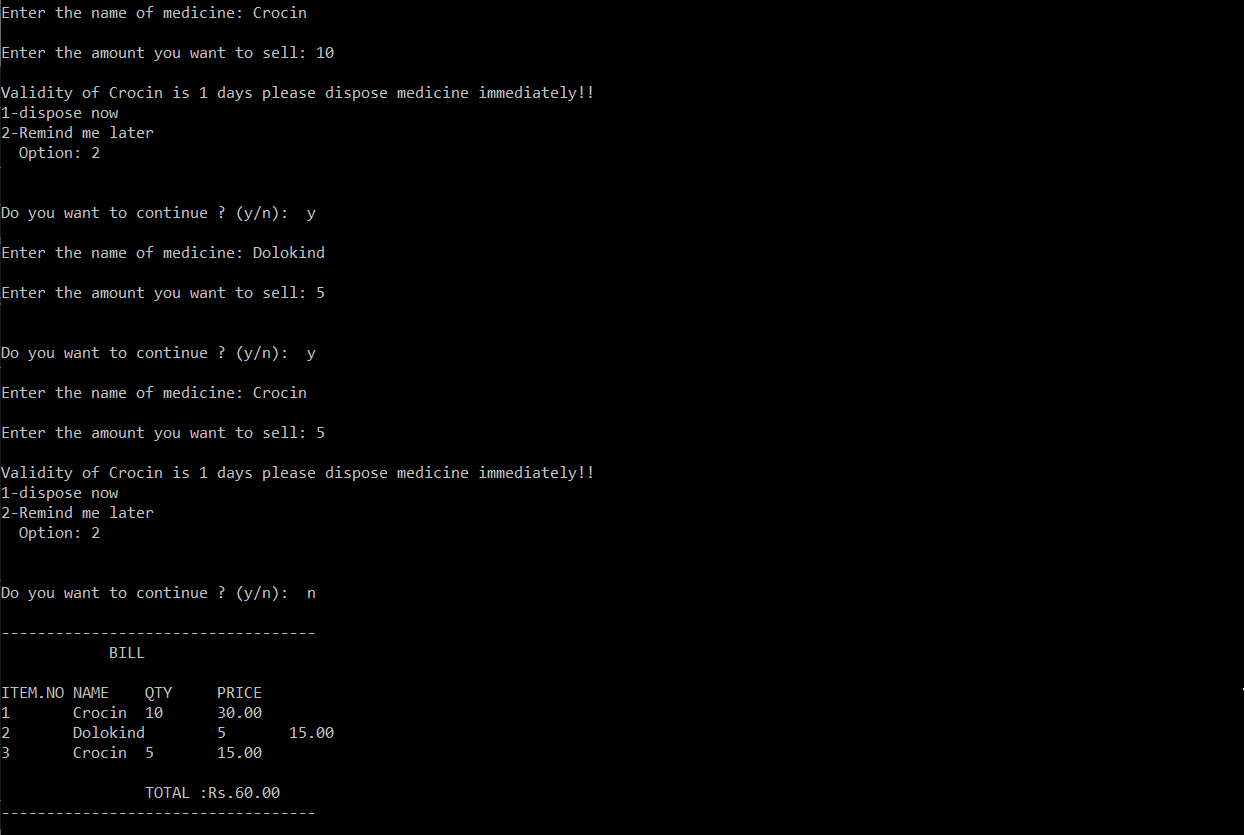
**Option 2: the stock for Crocin is updated and added to queue based on its expiry date**



**Option 4: details are displayed of all the existing medicines available in the drugstore along with all their attributes**



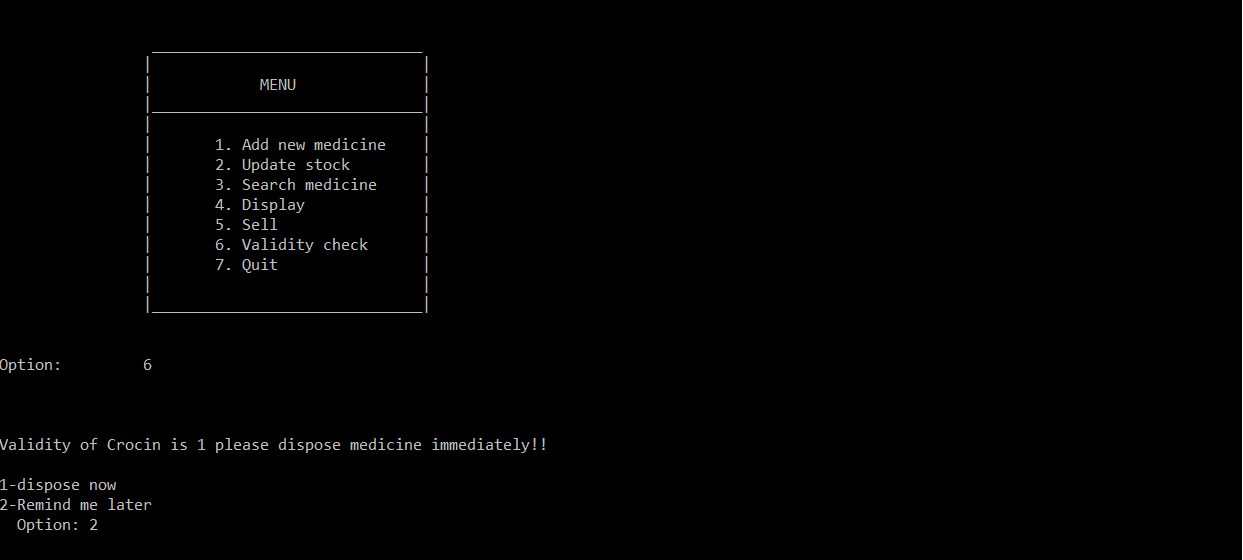
**Option 3: this is used to search for a medicine and the consequently, the details are displayed – here, Crocin is searched for**



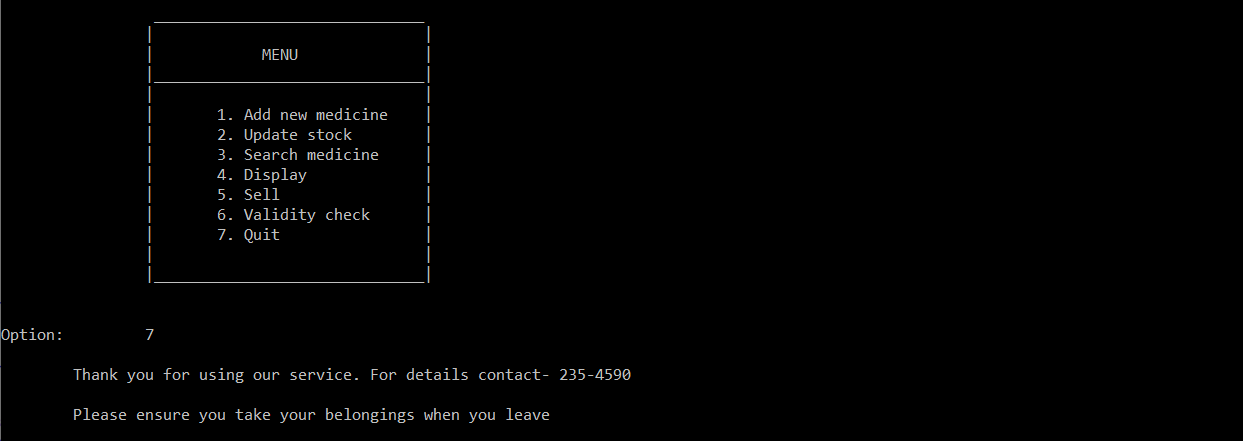
**Option 5: drugs are sold as the drug name and quantity are entered and subsequently, the bill is generated – here, Crocin and Dolokind are bought**



**Option 4: the details of the available medicines are displayed**



**Option 6: performs a validity check on the existing stock of medicines and displays alert if the medicine is about to expire soon – here, Crocin expires in 1 day and hence it can either be disposed right away or a reminder can be sent in later**



**Option 7: exits the program**

**Conclusion:**

Hence, a real-world problem such as drugstore management has been designed and implemented using data structures. The use of data structures has enhanced the way in which the operations are done and has provided additional functionality to the application.

**SOURCE CODE:**

**tree.h:**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<ctype.h>

#include <time.h>

#include<math.h>

typedef struct tree

{

int id;

char name[50];

char seller[50];

int total;

float cost;

queue \*q;

struct tree \*right, \*left;

}tree;

queue\* give\_queue(tree\* t, char name[])

{

if(t==NULL)

{

printf("\n\nError!");

return NULL;

}

if(strcmp(t->name,name)==0)

{

return t->q;

}

if(strcmp(name,t->name)<0)

give\_queue(t->left,name);

else if(strcmp(name,t->name)>0)

give\_queue(t->right,name);

}

float give\_cost(tree\* t,char name[])

{

if(t==NULL)

{

printf("Not found");

return 0;

}

if(strcmp(t->name,name)==0)

{

return t->cost;

}

if(strcmp(name,t->name)<0)

give\_cost(t->left,name);

else if(strcmp(name,t->name)>0)

give\_cost(t->right,name);

}

tree\* givetree(tree \*t,char name[])

{

if(t==NULL)

{

printf("Not found");

return 0;

}

if(strcmp(t->name,name)==0)

{

return t;

}

if(strcmp(name,t->name)<0)

give\_cost(t->left,name);

else if(strcmp(name,t->name)>0)

give\_cost(t->right,name);

}

void cal\_tot(tree \*t)

{

if(t!=NULL)

{

t->total=0;

for(int i=t->q->front;i<=t->q->rear;i++)

t->total+=t->q->qty[i];

}

}

tree\* insert(tree \*t,tree \*temp)

{

if(t==NULL)

{

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

//printf("Enter manufacture date(dd/mm/yyy) : ");

//scanf("%s",mfg);

//printf("Enter expiry date(dd/mm/yyyy) : ");

//scanf("%s",exp);

if(t==NULL)

printf("Out of space");

else

{

t->id=temp->id;

strcpy(t->name,temp->name);

strcpy(t->seller,temp->seller);

t->total=temp->total;

t->cost=temp->cost;

//strcpy(t->mfg,mfg);

//strcpy(t->exp,exp);

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

t->q=createQueue(20);

enqueue(t->q,t->total);

}

}

else if(strcmp(temp->name,t->name)<0){

//printf("Added to left!\n");

t->left=insert(t->left,temp);

}

else if(strcmp(temp->name,t->name)>0){

t->right=insert(t->right,temp);

//printf("Added to Right!\n");

}

return t;

}

void display(tree \*t)

{

if(t==NULL)

return;

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

printf("ID :%d\n",t->id);

printf("Name :%s\n",t->name);

printf("Seller :%s\n",t->seller);

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

printf("Price/tab :Rs.%.2f\n",t->cost);

printf("Stock :\n");

if(t->total>0)

{

printf(" Quantity Manufacture Date Expiry Date Validity(days)\n");

for(int i=t->q->front;i<=t->q->rear;i++)

{

printf("Batch %2d\t",i+1);

printf("%3d\t",t->q->qty[i]);

printf(" %2d/%2d/%4d\t",t->q->manf[i].dd, t->q->manf[i].mm, t->q->manf[i].yy);

printf(" %2d/%2d/%4d\t",t->q->exp[i].dd, t->q->exp[i].mm, t->q->exp[i].yy);

printf(" %4d\n",t->q->validity[i]);

}

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

}

//printf("Manufacture date :%s",t->mfg);

//printf("Expiry date :%s",t->exp);

}

int check(tree \*t, int a)

{

int f=-1;

if(t!=NULL)

{

for(int i=t->q->front;i<=t->q->rear;i++)

{

if(t->q->validity[i]<20)

{

printf("\nValidity of %s is %d days please dispose medicine immediately!!",t->name,t->q->validity[i]);

int ch;

printf("\n1-dispose now\n2-Remind me later\n Option: ");

scanf("%d",&ch);

if(ch==1)

{

printf("\nDisposing medicine....");

int temp=t->q->qty[i];

if(i==t->q->front){

t->q->front = (t->q->front + 1)%t->q->capacity;

}

else{

t->q->qty[i]=t->q->qty[i+1];

t->q->validity[i]=t->q->validity[i+1];

t->q->manf[i]=t->q->manf[i+1];

t->q->exp[i]=t->q->exp[i+1];

t->q->rear=(t->q->rear-1)%t->q->capacity;

}

t->q->size = t->q->size - 1;

t->total-=temp;

return -1;

}

}

}

}

if(t->total<a)

{

printf("\nNot enough stock!! ");

return -1;

}

else

{

return 1;

}

}

int warn(tree \*t)

{

int f=1;

if(t!=NULL)

{

for(int i=t->q->front;i<=t->q->rear;i++)

{

if(t->q->validity[i]<20)

{

f=-1;

printf("\nValidity of %s is %d please dispose medicine immediately!!",t->name,t->q->validity[i]);

int ch;

printf("\n\n1-dispose now\n2-Remind me later\n Option: ");

scanf("%d",&ch);

if(ch==1)

{

printf("\n\nDisposing medicine....\n");

int temp=t->q->qty[i];

if(i==t->q->front){

t->q->front = (t->q->front + 1)%t->q->capacity;

}

else{

t->q->qty[i]=t->q->qty[i+1];

t->q->validity[i]=t->q->validity[i+1];

t->q->manf[i]=t->q->manf[i+1];

t->q->exp[i]=t->q->exp[i+1];

t->q->rear=(t->q->rear-1)%t->q->capacity;

}

t->q->size = t->q->size - 1;

t->total-=temp;

}

}

warn(t->left);

warn(t->right);

}

}

return f;

}

void search(tree \*t,char name[50])

{

if(t==NULL)

{

printf("Not found");

return;

}

if(strcmp(t->name,name)==0)

{

display(t);

return;

}

if(strcmp(name,t->name)<0)

search(t->left,name);

else if(strcmp(name,t->name)>0)

search(t->right,name);

}

void inorder(tree \*t)

{

if(t==NULL)

return;

inorder(t->left);

display(t);

inorder(t->right);

}

**Queue.h:**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<ctype.h>

#include <time.h>

#include<math.h>

struct Date

{

int dd,mm,yy;

};

typedef struct Date Date;

typedef struct queue

{

int \*validity;

Date manf[20],exp[20];

int \*qty;

int front,rear,size,capacity;

}queue;

int dater(int x)

{ int y=0;

switch(x)

{

case 1: y=0; break;

case 2: y=31; break;

case 3: y=59; break;

case 4: y=90; break;

case 5: y=120;break;

case 6: y=151; break;

case 7: y=181; break;

case 8: y=212; break;

case 9: y=243; break;

case 10:y=273; break;

case 11:y=304; break;

case 12:y=334; break;

default: printf("Invalid Inputnnnn"); exit(1);

}

return(y);

}

int validate(struct Date m, struct Date e)

{

int ref,dd1,dd2,i,vald;

ref = m.yy;

dd1=0;

dd1=dater(m.mm);

for(i=ref;i<m.yy;i++)

{

if(i%4==0)

dd1+=1;

}

dd1=dd1+m.dd+(m.yy-ref)\*365;

dd2=0;

for(i=ref;i<e.yy;i++)

{

if(i%4==0)

dd2+=1;

}

dd2=dater(e.mm)+dd2+e.dd+((e.yy-ref)\*365);

vald=abs(dd2-dd1);

return vald;

}

queue\* createQueue(int maxelements)

{

queue\* q = (queue\*) malloc(sizeof(queue));

q->capacity = maxelements;

q->front = q->size = 0;

q->rear = maxelements - 1;

q->qty = (int\*) malloc(q->capacity \* sizeof(int));

q->validity = (int\*)malloc(q->capacity \* sizeof(int));

return q;

}

int isFull(queue\* q)

{

return (q->size == q->capacity);

}

int isEmpty(queue\* q)

{

return (q->size == 0);

}

void enqueue(queue\* q,int total)

{

if (isFull(q))

return;

q->rear = (q->rear + 1)%q->capacity;

//printf("Enter quantity: ");

q->qty[q->rear]=total;

//printf("Enter quantity: ");

//scanf("%d",&q->qty[q->rear]);

printf("\nEnter manufacture date(dd mm yyyy) : ");

scanf("%d %d %d",&q->manf[q->rear].dd,&q->manf[q->rear].mm,&q->manf[q->rear].yy);

printf("Enter expiry date (dd mm yyyy): ");

scanf("%d %d %d",&q->exp[q->rear].dd,&q->exp[q->rear].mm,&q->exp[q->rear].yy);

time\_t t = time(NULL);

struct tm tm = \*localtime(&t);

Date to ;

to.yy=tm.tm\_year + 1900;

to.mm= tm.tm\_mon + 1;

to.dd=tm.tm\_mday;

q->validity[q->rear]=validate(to,q->exp[q->rear]);

q->size = q->size + 1;

}

void sell(queue \*q, int amt)

{

int qt;

qt=amt;

/\*if(q->validity[q->front]<10)

{

printf("\nMedicine valid only %d days",q->validity[q->front]);

printf("\nDisposing medicine....");

q->front = (q->front + 1)%q->capacity;

q->size = q->size - 1;

printf("\nTry again...");

return;

}\*/

if(q->qty[q->front]<amt){

int t;

t=q->qty[q->front];

q->front = (q->front + 1);

q->size = q->size - 1;

sell(q,amt-t);

}

else if(q->qty[q->front]==amt)

{

q->front = (q->front + 1);

q->size = q->size - 1;

}

else{

q->qty[q->front]-=amt;

}

}

void disp(queue \*q)

{

for(int i=q->front;i<=q->rear;i++)

{

printf("Quantity : %d Manufacture date : %d-%d-%d Expiry date : %d-%d-%d Validity : %d\n",q->qty[i],q->manf[i].dd,q->manf[i].mm,q->manf[i].yy,q->exp[i].dd,q->exp[i].mm,q->exp[i].yy,q->validity[i]);

}

}

**menu.c:**

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<ctype.h>

#include <time.h>

#include<math.h>

#include"queue1.h"

#include"tree1.h"

void disp\_menu()

{

printf("\t\t");

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

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

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

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

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

printf("|\t1. Add new medicine |\n\t\t");

printf("|\t2. Update stock |\n\t\t");

printf("|\t3. Search medicine |\n\t\t");

printf("|\t4. Display |\n\t\t");

printf("|\t5. Sell |\n\t\t");

printf("|\t6. Validity check |\n\t\t");

printf("|\t7. Quit |\n\t\t");

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

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

}

int main()

{

tree \*t,\*temp;

t=NULL;

temp=(tree\*)malloc(sizeof(tree));

char name[50];

int o;

int qt;

printf("==========================================================================================================================================");

printf("\n\t\t\t\t\t\tWELCOME TO ABC PHARMACEUTICALS\n");

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

while(1)

{

printf("\n\n");

disp\_menu();

printf("\n\nOption: ");

scanf("%d",&o);

switch(o)

{

case 1:

printf("\n\nAdding medicine.....\n\n");

printf("\nEnter id : ");

scanf("%d",&temp->id);

printf("Enter name : ");

scanf("%s",temp->name);

printf("Enter seller : ");

scanf("%s",temp->seller);

printf("Enter quantity(tabs): ");

scanf("%d",&temp->total);

printf("Price/tab : ");

scanf("%f",&temp->cost);

t=insert(t,temp);

printf("MEDICINE ADDED!\n");

break;

case 2:

printf("\n\nUpdating stock....\n\n");

printf("Enter the name of medicine: ");

scanf("%s",name);

printf("\nEnter the amount you want to buy: ");

scanf("%d",&qt);

if(give\_queue(t,name)==NULL)

{

printf("\n\nMedicine not found!!");

continue;

}

enqueue(give\_queue(t,name),qt);

cal\_tot(givetree(t,name));

printf("\nSTOCK UPDATED!\n");

break;

case 3:

printf("\n\nSearch Results...\n\n");

printf("\nEnter name to search: ");

scanf("%s",name);

if(give\_queue(t,name)==NULL)

{

printf("\n\nMedicine not found!!");

continue;

}

search(t,name);

break;

case 4:

inorder(t);

break;

case 5:

printf("\n\nSelling Medicine....\n\n");

char ch;

char temp[15][15];

float price[15],tot=0;

int q[15],i=0;

do{

printf("\nEnter the name of medicine: ");

scanf("%s",name);

strcpy(temp[i],name);

printf("\nEnter the amount you want to sell: ");

scanf("%d",&qt);

q[i]=qt;

if(give\_queue(t,name)==NULL)

{

printf("\n\nMedicine not found!!");

continue;

}

int r=check(givetree(t,name),qt);

if(r!=-1){

sell(give\_queue(t,name),qt);

cal\_tot(givetree(t,name));

float c;

c=give\_cost(t,t->name);

c\*=qt;

price[i]=c;

tot+=price[i];

i++;

}

printf("\n\nDo you want to continue ? (y/n): ");

scanf(" %c",&ch);

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

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

printf("\t BILL\n\nITEM.NO\tNAME\tQTY\tPRICE\n");

for(int j=0;j<i;j++){

printf("%d\t%s\t%d\t%.2f\n",j+1,temp[j],q[j],price[j]);

}

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

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

break;

case 6:

printf("\n\n");

int f=warn(t);

if(f==1)

printf("All medicines are valid!\n");

break;

case 7:

printf("\n\tThank you for using our service. For details contact- 235-4590 \n\n\tPlease ensure you take your belongings when you leave\n");

exit(0);

}

}

return 0;

}