**G.H Raisoni College of Engineering & Management,Pune**

(An Autonomous Institute Affiliated to SPPU)

**Department: S.Y. Computer Engineering**

**Subject**: **(Data Structures)**

**Topic:-*Stationary Distribution system***

Submitted To:

Prof.Pournima More

Submitted by :

Amit Tiwari (SCOA01)

VidhyasagarBabalsure(SCOA02)

1. **Introduction:-**

Stationery distribution software refers to software which manages everything from keeping records, creating, adding or inserting data or information, deleting data/information, and many more. This software is made to minimize the human work and time of writing, remembering, sorting, calculating, etc.

****

**1.2 Objective:-**

\* The key objective of this is to specify the software

requirements and to explain more about stationary distribution system.

\* The idea and to help fellow administrators,

developers and the users of this Stationary distribution

System.

\* Plans, executes, and validates results of unit, systems,

and acceptance testing to ensure conformity to user\'s

requirements.

**1.3 Scope:-**

Stationary distribution system for general item shop application is intended to provide complete solutions for admin/shopkeepers, vendors as well as customers through a efficient way. The administration module will enable a system administrator and maintain various lists of stationaries category.

This system gives the benefits to shopkeeper for maintain the record and consume less time . It can be used to easily get the data and simple user interface helps to save time.

**2. Abstract:-**

This proposed system can be used by any new user and it does not require any educational level ,experience or technical expertise in computer field but it will be of good use if user has the good knowledge of how to operate a computer.

Stationary distribution system application enables shopkeepers to use and save the information, customers will get receipts for whatever they have taken, and maintain lists of shop categories. Also the Developer manage the items in the shop and also help admin/shopkeeper to see what items are available and in what quantity. They can easily see whatever the items are sold. Admin can sort items and see the price of each item as remembering prices of plenty of items is a difficult task.

**Software Requirements:-**

**Software Interface:-**

1. Operating System: Windows XP or above.

2. Linux x64/x86, Fedora or Ubuntu.

3. Turbo C or codeblocks with a gcc compiler.

**Hardware Interface:-**

Hardware requirements for insurance on internet will be same for both parties which are as follows:

1.Processor:Pentium.

2.RAM:256 MB

3.Hard Disk:160 MB

**Advantages:-**

****

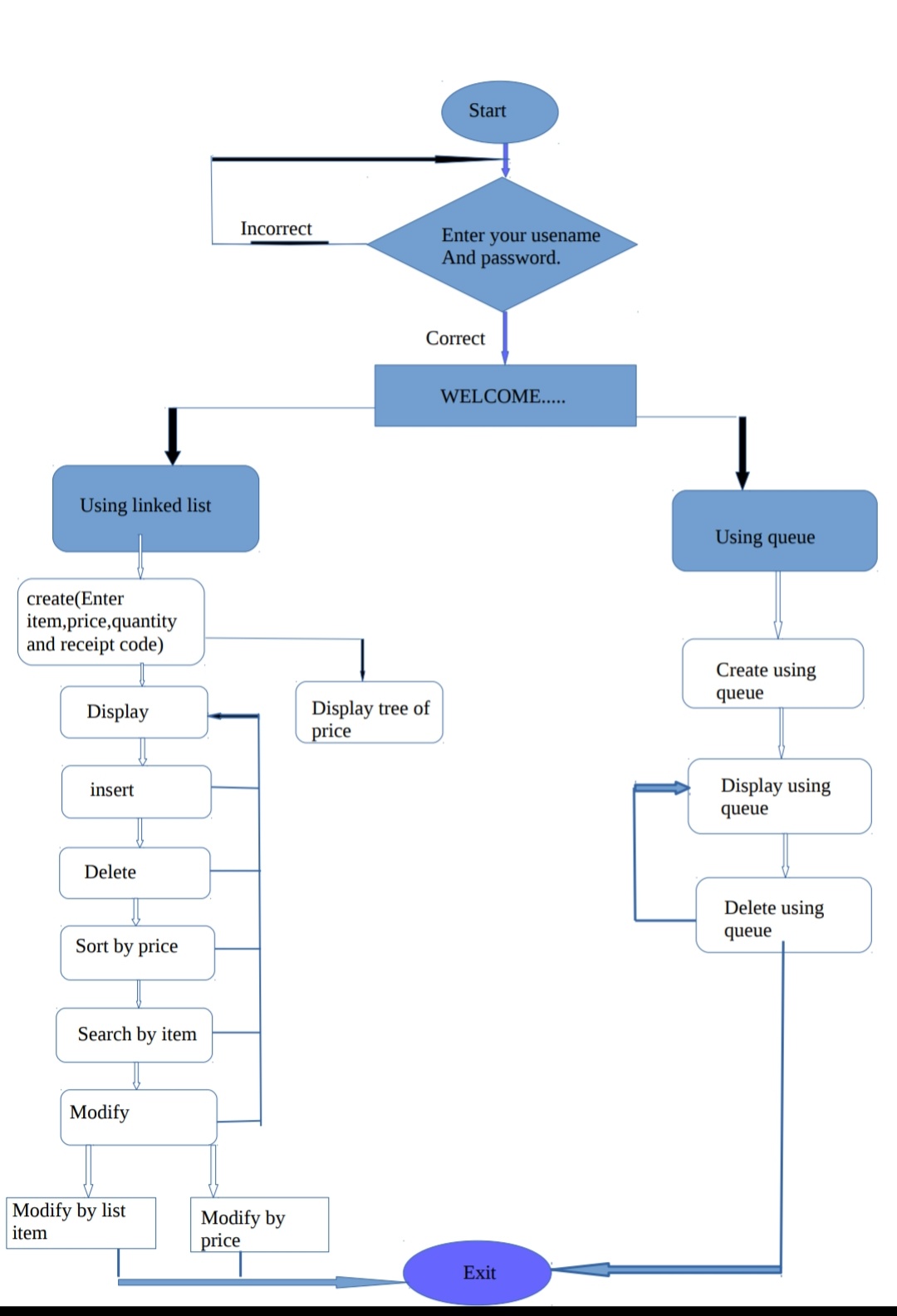
* Easy to handle.
* Reduce the work.
* Less time consuming.
* User friendly.
* Low start-up cost and maintenance.
* No software maintenance.

**Drawbacks:-**

* Limited design freedom on some solutions
* Limited customization and fixed memory needed.
* Maintain cost.
* Little knowledge of computer required.

**CONCLUSION:-**

This system is the best way to shopkeeper to keep records of stationary products. Due to this system shopkeeper don't need to search item here and there in shop instead that shopkeeper can find out the appropriate position of item .This software enables user to create or add a new item in stationary list. Also user can insert, modify(update) or delete items from the list of items. The software made is simple to use and it is user friendly as it clearly asks which operation to perform.



**(DS 9th (a):-)**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define COUNT 10

struct stationary

{

char list[20];

int price;

int quantity;

int receipt\_code;

struct stationary \*link;

}\*start;

struct priceTree

{

int price;

struct priceTree \*left;

struct priceTree \*right;

};

struct stationary\* getNewNode();

struct stationary \*create(struct stationary \*start);

void display (struct stationary \*start);

struct stationary \*insert(struct stationary \*start);

struct stationary \*del(struct stationary \*start,char item[]);

struct stationary \*sort (struct stationary \*start);

void \*search(struct stationary \*start,char item[]);

struct stationary \*modify(struct stationary \*start,char item[]);

void create\_queue();

void display\_queue();

void delete\_queue();

struct stationary \*front=NULL;

struct stationary \*rear=NULL;

struct priceTree \*node;

struct priceTree \*root;

struct priceTree \*getTree();

void sortTree(int arr[],int n);

struct priceTree \*getnode(int x);

struct priceTree \*BalancedBst(int arr[],int s,int e);

void print2DUtil(struct priceTree \*root, int space);

void print2D(struct priceTree \*root);

int i,n,x;

int s,e;

void main()

{

char item[20];

int ch;

struct stationary \*start=NULL;

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

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

printf("\t\t\*\* STATIONARY DISTRIBUTION SYSTEM \*\*\n");

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

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

printf("1) to create\n");

printf("2) to display\n");

printf("3) to insert\n");

printf("4) to delete\n");

printf("5) to sort by price\n");

printf("6) to search\n");

printf("7) to modify:\n");

printf("8)create using queue.\n");

printf("9) display using queue.\n");

printf("10) delete using queue.\n");

printf("11) create tree. \n");

printf("12)exit(0)\n" );

while(1)

{

printf("Enter choice-");

scanf("%d",&ch);

switch (ch)

{

case 1:

start=create(start);

break;

case 2:

display(start);

break;

case 3:

start=insert(start);

break;

case 4:

printf("Enter item you want to delete:");

scanf("%s",item);

start=del(start,item);

break;

case 5:

sort(start);

break;

case 6:

printf("Enter item you want to search:");

scanf("%s",item);

search(start,item);

break;

case 7:

printf("Enter item you want to modify:");

scanf("%s",item);

start=modify(start,item);

break;

case 8:

create\_queue();

break;

case 9:

display\_queue();

break;

case 10:

delete\_queue();

break;

case 11:

print2D(getTree(start));

break;

case 12:

exit(0);

break;

default:

printf("error!");

}

}

}

struct stationary \*create(struct stationary \*start)

{

struct stationary \*p,\*temp;

printf("Enter no of type of items:");

scanf("%d",&n);

temp=start;

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

{ temp=(struct stationary\*)malloc(sizeof(struct stationary)\*1);

printf("\tenter product name:");

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

printf("\tenter price:");

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

printf("\tenter quantity:");

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

printf("\tenter receipt code:");

scanf("%d",&(temp->receipt\_code));

printf("\n");

if(start==NULL)

{

start=temp;

temp->link=NULL;

}

else

{

p=start;

while(p->link!=NULL)

p=p->link;

p->link=temp;

temp->link=NULL;

}

}

return start;

}

void display (struct stationary \*start)

{

struct stationary \*p;

if(start==NULL)

{

printf("\n\*Order List is Empty\*\n\n");

return;

}

p=start;

printf("\n\tItem\tPrice\tquantity\treceipt no.\n");

while(p!=NULL)

{

printf("\n\t%s\t %d\t%d\t\t%d",p->list,p->price,p->quantity,p->receipt\_code);

p=p->link;

}

printf("\n");

}

struct stationary \*insert(struct stationary \*start)

{

int pos;

struct stationary \*p,\*temp;

temp=(struct stationary\*)malloc(sizeof(struct stationary)\*1);

printf("Enter location you want to insert:");

scanf("%d",&pos);

if(pos==1)

{

printf("\tenter product name:");

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

printf("\tenter price:");

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

printf("\tenter quantity of product:");

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

printf("\tenter receipt no. of product");

scanf("%d",&(temp->receipt\_code));

temp->link=start;

start=temp;

return start;

}

p=start;

for(i=1;i<pos-1 && p!=NULL;i++)

p=p->link;

if(p==NULL)

printf("\n\*less no of items\*\n\n");

else

{ printf("\tenter product name:");

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

printf("\tenter price:");

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

printf("\tenter quantity of product:");

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

printf("\tenter receipt no. of product");

scanf("%d",&(temp->receipt\_code));

temp->link=p->link;

p->link=temp;

}

return start;

}

struct stationary \*del(struct stationary \*start,char item[])

{

struct stationary \*temp,\*p;

if(strcmp(start->list,item)==0)

{

temp=start;

start=temp->link;

free(temp);

return start;

}

p=start;

while(p->link!=NULL)

{

if(strcmp(p->link->list,item)==0)

{

temp=p->link;

p->link=temp->link;

free(temp);

return start;

}

p=p->link;

}

}

struct stationary \*sort (struct stationary \*start)

{

struct stationary \*p,\*q,\*temp;

temp=(struct stationary\*)malloc(sizeof(struct stationary)\*1);

for(p=start;p->link!=NULL;p=p->link)

{

for(q=p->link;q!=NULL;q=q->link)

{

if(p->price > q->price)

{

temp->price=p->price;

p->price=q->price;

q->price=temp->price;

strcpy(temp->list,p->list);

strcpy(p->list,q->list);

strcpy(q->list,temp->list);

}

}

}

}

void \*search(struct stationary \*start,char item[])

{

struct stationary \*p;

int flag=0;

if(strcmp(start->list,item)==0)

{

printf("\nITEM FOUND\n");

printf("\nItem\tPrice\tquantity\treceipt\_no.");

printf("\n%s\t %d\t %d\t %d\n",start->list,start->price,start->quantity,start->receipt\_code);

flag++;

}

p=start;

while(p->link!=NULL)

{

if(strcmp(p->link->list,item)==0)

{

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

printf("\nItem\tPrice\tquantity\treceipt\_no.");

printf("\n%s\t %d\t %d\t %d\n",p->link->list,p->link->price,p->link->quantity,p->link->receipt\_code);

}

p=p->link;

flag++;

}

if(flag==0)

printf("\n\nnot found\n\n");

}

struct stationary \*modify(struct stationary \*start,char item[])

{

struct stationary \*p;

int choice;

char ans1,ans2;

int flag=0,chw=0;

if(strcmp(start->list,item)==0)

{

printf("\nITEM FOUND\n");

printf("\nItem\tPrice\tquantity\treceipt\_no.");

printf("\n%s\t %d\t %d\t %d\n",start->list,start->price,start->quantity,start->receipt\_code);

do

{

printf("enter:\n 1) to modify list name\n 2) to modify list price:");

scanf("%d",&choice);

switch (choice)

{

case 1:

printf("enter new product name:");

scanf("%s",(start->list));

break;

case 2:

printf("enter new product price:");

scanf("%d",&(start->price));

break;

}

printf("press 1 to continue modifying:");

scanf("%d",&chw);

}while(chw==1);

flag++;

return start;

}

p=start;

while(p->link!=NULL)

{

if(strcmp(p->link->list,item)==0)

{

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

printf("\nItem\tPrice\tquantity\treceipt\_no.");

printf("\n%s\t %d\t %d\t %d\n",p->link->list,p->link->price,p->link->quantity,p->link->receipt\_code);

do

{

printf("enter:\n 1) to modify list name\n 2) to modify list price:");

scanf("%d",&choice);

switch (choice)

{

case 1:

printf("enter new product name:");

scanf("%s",(p->link->list));

break;

case 2:

printf("enter new product price:");

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

break;

}

printf("press 1 to continue modifying:");

scanf("%d",&chw);

}while(chw==1);

}

p=p->link;

flag++;

return start;

}

if(flag==0)

printf("\n\nnot found\n\n");

}

struct stationary\* getNewNode()

{

return (struct stationary\*)malloc(sizeof(struct stationary));

}

void create\_queue()

{

struct stationary \*temp;

int n;

printf("Enter no.of type of item:");

scanf("%d",&n);

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

{

temp=getNewNode();

printf("\tEnter product name:");

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

printf("\tEnter price:");

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

printf("\tEnter quantity:");

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

printf("\tEnter receipt\_code:");

scanf("%d",&(temp->receipt\_code));

if (rear==NULL)

{

rear= temp;

front = temp;

rear->link=NULL;

continue;

}

rear->link=temp;

rear=temp;

}

}

void display\_queue()

{

struct stationary \*p;

if(front==NULL)

{

printf("\n No ITEM IN LIST\n");

}

else

{

p=front;

printf("\n\tLIST \tPRICE \tQUANTITY \tRECEIPT CODE \n\n");

while(p!=NULL)

{

printf("\t%s\t%d\t%d\t%d",p->list,p->price,p->quantity,p->receipt\_code);

printf("\n");

p=p->link;

}

}

}

void delete\_queue()

{

struct stationary \*temp;

if(front==NULL)

{

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

}

else

{

temp=front;

front=front->link;

free(temp);

}

}

struct priceTree \*getTree(struct stationary \*start)

{

int arr[n];

int point=0;

struct stationary \*temp=start;

while(temp!=NULL)

{

arr[point++]=temp->price;

temp=temp->link;

}

sortTree(arr,n);

return BalancedBst(arr,0,n-1);

}

void sortTree(int arr[],int n)

{

int temp,i,j;

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

{

for(int j = 0; j <= (n-1-i); j++)

{

if(arr[j] > arr[j+1]) //for decending order change "arr[j]<arr[j+1]"

{

temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

}

struct priceTree \*getnode(int x)

{

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

temp->price=x;

temp->left=temp->right=NULL;

return temp;

}

struct priceTree \*BalancedBst(int arr[],int s,int e)

{

if(s > e)

return NULL;

int mid=(s+e)/2;

struct priceTree \*root=getnode(arr[mid]);

root->left=BalancedBst(arr,s,mid-1);

root->right=BalancedBst(arr,mid+1,e);

return root;

}

void print2DUtil(struct priceTree \*root, int space)

{

if (root == NULL)

return;

space+=COUNT;

print2DUtil(root->right, space);

printf("\n");

for (int i = COUNT; i < space; i++)

printf(" ");

printf("%d\n\n", root->price);

print2DUtil(root->left, space);

}

void print2D(struct priceTree \*root)

{

print2DUtil(root, 0);

}

/\* OUTPUT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* STATIONARY DISTRIBUTION SYSTEM \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1) to create

2) to display

3) to insert

4) to delete

5) to sort by price

6) to search

7) to modify:

8)create using queue.

9) display using queue.

10) delete using queue.

11) create tree.

12)exit(0)

Enter choice-1

Enter no of type of items:7

enter product name:book

enter price:50

enter quantity:25

enter receipt code:1

enter product name:pen

enter price:10

enter quantity:40

enter receipt code:2

enter product name:marker

enter price:25

enter quantity:20

enter receipt code:3

enter product name:file

enter price:20

enter quantity:40

enter receipt code:4

enter product name:notebook

enter price:45

enter quantity:20

enter receipt code:5

enter product name:paper

enter price:5

enter quantity:22

enter receipt code:6

enter product name:bookbinding

enter price:100

enter quantity:21

enter receipt code:7

Enter choice-2

Item Price quantity receipt no.

book 50 25 1

pen 10 40 2

marker 25 20 3

file 20 40 4

notebook 45 20 5

paper 5 22 6

bookbinding 100 21 7

Enter choice-11

100

50

45

25

20

10

5

Enter choice: \*/

**(DS 9th (B):-)**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#define COUNT 10

struct stationary

{

char list[20];

int price;

int quantity;

int receipt\_code;

struct stationary \*link;

}\*start;

struct priceTree

{

int price;

struct priceTree \*left;

struct priceTree \*right;

};

struct stationary\* getNewNode();

struct stationary \*create(struct stationary \*start);

struct stationary \*insert(struct stationary \*start,int n);

struct stationary \*del(struct stationary \*start,char item[]);

struct stationary \*sort (struct stationary \*start);

struct stationary \*modify(struct stationary \*start,char item[]);

void display (struct stationary \*start);

void \*search(struct stationary \*start,char item[]);

void create\_queue();

void display\_queue();

void delete\_queue();

struct stationary \*front=NULL;

struct stationary \*rear=NULL;

struct priceTree \*node;

struct priceTree \*root;

struct priceTree \*getTree();

struct priceTree \*getnode(int x);

struct priceTree \*BalancedBst(int arr[],int s,int e);

void sortTree(int arr[],int n);

void print2DUtil(struct priceTree \*root, int space);

void print2D(struct priceTree \*root);

struct stationary \*p,\*q,\*temp;

char item[20];

int i,n,x,j,ch,l;

int s,e,point=0;

int flag=0;

void main()

{

struct stationary \*start=NULL;

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

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

printf("\t\t\*\* STATIONARY DISTRIBUTION SYSTEM \*\*\n");

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

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

while(1)

{

printf(" 1) to create\n 2) to display.\n 3) to insert.\n 4) to delete. \n 5) to sort by price.\n 6) to search. \n 7) to modify. \n 8) create using queue.\n 9) display using queue.\n 10) delete using queue.\n 11) Create tree of price.\n 12) exit(0).\n\n ");

printf("Enter your choice-");

scanf("%d",&ch);

switch (ch)

{

case 1:

start=create(start);

break;

case 2:

display(start);

break;

case 3:

start=insert(start,n);

break;

case 4:

printf("Enter item you want to delete:");

scanf("%s",item);

start=del(start,item);

break;

case 5:

sort(start);

break;

case 6:

printf("Enter item you want to search:");

scanf("%s",item);

search(start,item);

break;

case 7:

printf("Enter item you want to modify:");

scanf("%s",item);

start=modify(start,item);

break;

case 8:

create\_queue();

break;

case 9:

display\_queue();

break;

case 10:

delete\_queue();

break;

case 11:

print2D(getTree(start));

break;

case 12:

exit(0);

break;

default:

printf("error!");

}

}

}

struct stationary \*create(struct stationary \*start)

{

struct stationary \*temp;

printf("Enter no of type of items:");

scanf("%d",&n);

temp=start;

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

{

temp=(struct stationary\*)malloc(sizeof(struct stationary)\*1);

while(1)

{

printf("\tenter product name(Length 1 to 10):");

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

l=strlen(temp->list);

if(l > 10 )

{

printf("\tPlease enter less characters in list name\n");

}

else

break;

}

do{

printf("\tenter price (1 to 999):");

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

if(temp->price > 1 && temp->price < 999)

{

break;

}

else

{

printf("\tPlease enter less price\n");

}

}while(1);

do{

printf("\tenter quantity(1 to 500):");

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

if(temp->quantity > 1 && temp->quantity < 500)

{

break;

}

else

{

printf("\tPlease enter less quantity\n");

}

}while(1);

do{

printf("\tenter receipt code(1 to 200):");

scanf("%d",&(temp->receipt\_code));

if(temp-> receipt\_code > 0 && temp-> receipt\_code < 200)

{

break;

}

else

{

printf("\tSORRY..not available Please enter receipt\_code between(1 to 200)\n");

}

}while(1);

printf("\n");

if(start==NULL)

{

start=temp;

temp->link=NULL;

}

else

{

p=start;

while(p->link!=NULL)

p=p->link;

p->link=temp;

temp->link=NULL;

}

}

return start;

}

void display (struct stationary \*start)

{

if(start==NULL)

{

printf("\n\*Order List is Empty\*\n\n");

return;

}

p=start;

printf("\n\tItem\tPrice\tquantity\treceipt no.\n");

while(p!=NULL)

{

printf("\n\t%s\t %d\t%d\t\t%d",p->list,p->price,p->quantity,p->receipt\_code);

p=p->link;

}

printf("\n");

}

struct stationary \*insert(struct stationary \*start,int n)

{

struct stationary \*temp;

int pos,count=0;

temp=(struct stationary\*)malloc(sizeof(struct stationary)\*1);

printf("Enter location you want to insert:");

scanf("%d",&pos);

if(pos<n)

{

count++;

}

n=n+count;

printf("%d\n",n );

if(pos==1)

{

while(1)

{

printf("\tenter product name(Length 1 to 10):");

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

l=strlen(temp->list);

if(l > 10 )

{

printf("\tPlease enter less characters in list name\n");

}

else

break;

}

do{

printf("\tenter price(1 to 999):");

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

if(temp->price > 1 && temp->price < 999)

{

break;

}

else

{

printf("\tPlease enter less price\n");

}

}while(1);

do{

printf("\tenter quantity(1 to 500):");

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

if(temp->quantity > 1 && temp->quantity < 500)

{

break;

}

else

{

printf("\tPlease enter less quantity\n");

}

}while(1);

do{

printf("\tenter receipt code(1 to 200):");

scanf("%d",&(temp->receipt\_code));

if(temp-> receipt\_code > 0 && temp-> receipt\_code < 200)

{

break;

}

else

{

printf("\tSORRY..not available Please enter receipt\_code between(1 to 200)\n");

}

}while(1);

temp->link=start;

start=temp;

return start;

}

p=start;

for(i=1;i<pos-1 && p!=NULL;i++)

p=p->link;

if(p==NULL)

printf("\n\*less no of items\*\n\n");

else

{

while(1)

{

printf("\tenter product name(Length 1 to 10):");

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

l=strlen(temp->list);

if(l > 10 )

{

printf("\tPlease enter less characters in list name\n");

}

else

break;

}

do{

printf("\tenter price(1 to 999):");

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

if(temp->price > 1 && temp->price < 999)

{

break;

}

else

{

printf("\tPlease enter less price\n");

}

}while(1);

do{

printf("\tenter quantity(1 to 500):");

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

if(temp->quantity > 1 && temp->quantity < 500)

{

break;

}

else

{

printf("\tPlease enter less quantity\n");

}

}while(1);

do{

printf("\tenter receipt code(1 to 200):");

scanf("%d",&(temp->receipt\_code));

if(temp-> receipt\_code > 0 && temp-> receipt\_code < 200)

{

break;

}

else

{

printf("\tSORRY..not available Please enter receipt\_code between(1 to 200)\n");

}

}while(1);

temp->link=p->link;

p->link=temp;

}

return start;

}

struct stationary \*del(struct stationary \*start,char item[])

{

struct stationary \*temp;

if(strcmp(start->list,item)==0)

{

temp=start;

start=temp->link;

free(temp);

return start;

}

p=start;

while(p->link!=NULL)

{

if(strcmp(p->link->list,item)==0)

{

temp=p->link;

p->link=temp->link;

free(temp);

return start;

}

p=p->link;

}

}

struct stationary \*sort (struct stationary \*start)

{

struct stationary \*temp;

temp=(struct stationary\*)malloc(sizeof(struct stationary)\*1);

for(p=start;p->link!=NULL;p=p->link)

{

for(q=p->link;q!=NULL;q=q->link)

{

if(p->price > q->price)

{

temp->price=p->price;

p->price=q->price;

q->price=temp->price;

strcpy(temp->list,p->list);

strcpy(p->list,q->list);

strcpy(q->list,temp->list);

}

}

}

}

void \*search(struct stationary \*start,char item[])

{

if(strcmp(start->list,item)==0)

{

printf("\nITEM FOUND\n");

printf("\nItem\tPrice\tquantity\treceipt\_no.");

printf("\n%s\t %d\t %d\t %d\n",start->list,start->price,start->quantity,start->receipt\_code);

flag++;

}

p=start;

while(p->link!=NULL)

{

if(strcmp(p->link->list,item)==0)

{

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

printf("\nItem\tPrice\tquantity\treceipt\_no.");

printf("\n%s\t %d\t %d\t %d\n",p->link->list,p->link->price,p->link->quantity,p->link->receipt\_code);

}

p=p->link;

flag++;

}

if(flag==0)

printf("\n\nnot found\n\n");

}

struct stationary \*modify(struct stationary \*start,char item[])

{

char ans1,ans2;

int chw=0;

if(strcmp(start->list,item)==0)

{

printf("\nITEM FOUND\n");

printf("\nItem\tPrice\tquantity\treceipt\_no.");

printf("\n%s\t %d\t %d\t %d\n",start->list,start->price,start->quantity,start->receipt\_code);

do

{

printf("enter:\n 1) to modify list name\n 2) to modify list price:");

scanf("%d",&ch);

switch (ch)

{

case 1:

while(1)

{

printf("\tenter product name(Length 1 to 10):");

scanf("%s",(start->list));

l=strlen(start->list);

if(l > 10 )

{

printf("\tPlease enter less characters in list name\n");

}

else

{

break;

}

}

break;

case 2:

do{

printf("\tenter price(1 to 999):");

scanf("%d",&(start->price));

if(start->price > 1 && start->price < 999)

{

break;

}

else

{

printf("\tPlease enter less price\n");

}

}while(1);

break;

}

printf("press 1 to continue modifying:");

scanf("%d",&chw);

}while(chw==1);

flag++;

return start;

}

p=start;

while(p->link!=NULL)

{

if(strcmp(p->link->list,item)==0)

{

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

printf("\nItem\tPrice\tquantity\treceipt\_no.");

printf("\n%s\t %d\t %d\t %d\n",p->link->list,p->link->price,p->link->quantity,p->link->receipt\_code);

do

{

printf("enter:\n 1) to modify list name\n 2) to modify list price:");

scanf("%d",&ch);

switch (ch)

{

case 1:

while(1)

{

printf("\tenter product name(Length 1 to 10):");

scanf("%s",(p->link->list));

l=strlen(p->link->list);

if(l > 10 )

{

printf("\tPlease enter less characters in list name\n");

}

else

{

break;

}

}

break;

case 2:

do{

printf("\tenter price(1 to 999):");

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

if(p->link->price > 1 && p->link->price < 999)

{

break;

}

else

{

printf("\tPlease enter less price\n");

}

}while(1);

break;

}

printf("press 1 to continue modifying:");

scanf("%d",&chw);

}while(chw==1);

}

p=p->link;

flag++;

return start;

}

if(flag==0)

printf("\n\nnot found\n\n");

}

struct stationary\* getNewNode()

{

return (struct stationary\*)malloc(sizeof(struct stationary));

}

void create\_queue()

{

int q;

struct stationary \*temp;

printf("Enter no.of type of item:");

scanf("%d",&q);

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

{

temp=getNewNode();

while(1)

{

printf("\tenter product name(Length 1 to 10):");

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

l=strlen(temp->list);

if(l > 10 )

{

printf("\tPlease enter less characters in list name\n");

}

else

{

break;

}

}

do{

printf("\tenter price(1 to 999):");

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

if(temp->price > 1 && temp->price < 999)

{

break;

}

else

{

printf("\tPlease enter less price\n");

}

}while(1);

do{

printf("\tenter quantity(1 to 500):");

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

if(temp->quantity > 1 && temp->quantity < 500)

{

break;

}

else

{

printf("\tPlease enter less quantity\n");

}

}while(1);

do{

printf("\tenter receipt code(1 to 200):");

scanf("%d",&(temp->receipt\_code));

if(temp-> receipt\_code > 0 && temp-> receipt\_code < 200)

{

break;

}

else

{

printf("\tSORRY..not available Please enter receipt\_code between(1 to 200)\n");

}

}while(1);

printf("\n");

if (rear==NULL)

{

rear= temp;

front = temp;

rear->link=NULL;

continue;

}

rear->link=temp;

rear=temp;

}

}

void display\_queue()

{

if(front==NULL)

{

printf("\n No ITEM IN LIST\n");

}

else

{

p=front;

printf("\n\tLIST \tPRICE \tQUANTITY \tRECEIPT CODE \n\n");

while(p!=NULL)

{

printf("\t%s\t%d\t%d\t%d",p->list,p->price,p->quantity,p->receipt\_code);

printf("\n");

p=p->link;

}

}

}

void delete\_queue()

{

if(front==NULL)

{

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

}

else

{

temp=front;

front=front->link;

free(temp);

}

}

struct priceTree \*getTree(struct stationary \*start)

{

int arr[n];

struct stationary \*temp=start;

while(temp!=NULL)

{

arr[point++]=temp->price;

temp=temp->link;

}

sortTree(arr,n);

return BalancedBst(arr,0,n-1);

}

void sortTree(int arr[],int n)

{

int temp;

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

{

for(int j = 0; j < (n-1-i); j++)

{

if(arr[j] > arr[j+1]) //for decending order change "arr[j]<arr[j+1]"

{

temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

}

struct priceTree \*getnode(int x)

{

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

temp->price=x;

temp->left=temp->right=NULL;

return temp;

}

struct priceTree \*BalancedBst(int arr[],int s,int e)

{

if(s > e)

return NULL;

int mid=(s+e)/2;

struct priceTree \*root=getnode(arr[mid]);

root->left=BalancedBst(arr,s,mid-1);

root->right=BalancedBst(arr,mid+1,e);

return root;

}

void print2DUtil(struct priceTree \*root, int space)

{

if (root == NULL)

return;

space+=COUNT;

print2DUtil(root->right, space);

printf("\n");

for (int i = COUNT; i < space; i++)

printf(" ");

printf("%d\n\n", root->price);

print2DUtil(root->left, space);

}

void print2D(struct priceTree \*root)

{

print2DUtil(root, 0);

}

Output:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* STATIONARY DISTRIBUTION SYSTEM \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-1

Enter no of type of items:3

enter product name(Length 1 to 10):book

enter price (1 to 999):50

enter quantity(1 to 500):25

enter receipt code(1 to 200):1

enter product name(Length 1 to 10):asdfghjkliuytre

Please enter less characters in list name

enter product name(Length 1 to 10):pen

enter price (1 to 999):15

enter quantity(1 to 500):20

enter receipt code(1 to 200):2

enter product name(Length 1 to 10):file

enter price (1 to 999):1254

Please enter less price

enter price (1 to 999):25

enter quantity(1 to 500):568

Please enter less quantity

enter quantity(1 to 500):52

enter receipt code(1 to 200):265

SORRY..not available Please enter receipt\_code between(1 to 200)

enter receipt code(1 to 200):3

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-2

Item Price quantity receipt no.

book 50 25 1

pen 15 20 2

file 25 52 3

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-3

Enter location you want to insert:2

4

enter product name(Length 1 to 10):pencil

enter price(1 to 999):8

enter quantity(1 to 500):666

Please enter less quantity

enter quantity(1 to 500):90

enter receipt code(1 to 200):412

SORRY..not available Please enter receipt\_code between(1 to 200)

enter receipt code(1 to 200):4

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-2

Item Price quantity receipt no.

book 50 25 1

pencil 8 90 4

pen 15 20 2

file 25 52 3

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-8

Enter no.of type of item:2

enter product name(Length 1 to 10):ertyuilkjhgfds

Please enter less characters in list name

enter product name(Length 1 to 10):Eraser

enter price(1 to 999):5

enter quantity(1 to 500):589

Please enter less quantity

enter quantity(1 to 500):100

enter receipt code(1 to 200):698

SORRY..not available Please enter receipt\_code between(1 to 200)

enter receipt code(1 to 200):5

enter product name(Length 1 to 10):chart

enter price(1 to 999):15

enter quantity(1 to 500):26

enter receipt code(1 to 200):6

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-9

LIST PRICE QUANTITY RECEIPT CODE

Eraser 5 100 5

chart 15 26 6

DS 10:

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<conio.h>

#define COUNT 10

struct stationary

{

char list[20];

int price;

int quantity;

int receipt\_code;

struct stationary \*link;

}\*start;

struct priceTree

{

int price;

struct priceTree \*left;

struct priceTree \*right;

};

void login();

struct stationary\* getNewNode();

struct stationary \*create(struct stationary \*start);

struct stationary \*insert(struct stationary \*start);

struct stationary \*del(struct stationary \*start,char item[]);

struct stationary \*sort (struct stationary \*start);

struct stationary \*modify(struct stationary \*start,char item[]);

void display (struct stationary \*start);

void \*search(struct stationary \*start,char item[]);

void create\_queue();

void display\_queue();

void delete\_queue();

struct stationary \*front=NULL;

struct stationary \*rear=NULL;

struct priceTree \*node;

struct priceTree \*root;

struct priceTree \*getTree();

struct priceTree \*getnode(int x);

struct priceTree \*BalancedBst(int arr[],int s,int e);

void sortTree(int arr[],int n);

void print2DUtil(struct priceTree \*root, int space);

void print2D(struct priceTree \*root);

struct stationary \*p,\*q,\*temp;

char item[20];

char name[20]="stationary";

char password[5]="sagar";

char pass[4];

int i,n,x,j,ch,l;

int s,e,point=0;

int flag=0;

void main()

{

login();

struct stationary \*start=NULL;

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

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

printf("\t\t\*\* STATIONARY DISTRIBUTION SYSTEM \*\*\n");

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

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

while(1)

{

printf(" 1) to create\n 2) to display.\n 3) to insert.\n 4) to delete. \n 5) to sort by price.\n 6) to search. \n 7) to modify. \n 8) create using queue.\n 9) display using queue.\n 10) delete using queue.\n 11) Create tree of price.\n 12) exit(0).\n\n ");

printf("Enter your choice-");

scanf("%d",&ch);

switch (ch)

{

case 1:

start=create(start);

break;

case 2:

display(start);

break;

case 3:

start=insert(start);

break;

case 4:

printf("Enter item you want to delete:");

scanf("%s",item);

start=del(start,item);

break;

case 5:

sort(start);

break;

case 6:

printf("Enter item you want to search:");

scanf("%s",item);

search(start,item);

break;

case 7:

printf("Enter item you want to modify:");

scanf("%s",item);

start=modify(start,item);

break;

case 8:

create\_queue();

break;

case 9:

display\_queue();

break;

case 10:

delete\_queue();

break;

case 11:

print2D(getTree(start));

break;

case 12:

exit(0);

break;

default:

printf("error!");

}

}

}

void login()

{

printf("\t\tEnter username:");

scanf("%s",name);

printf("\t\tEnter the password:");

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

i=0;

while(ch!=13)

{

ch=getch();

pass[i]=ch;

i++;

printf("\*");

}

pass[i-1]='\0';

if(strcmp(password,pass)==0)

{

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

}

else

{

printf("\nYour system has been locked");

exit (0);

}

}

struct stationary \*create(struct stationary \*start)

{

struct stationary \*temp;

printf("Enter no of type of items:");

scanf("%d",&n);

temp=start;

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

{

temp=(struct stationary\*)malloc(sizeof(struct stationary)\*1);

while(1)

{

printf("\tenter product name(Length 1 to 10):");

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

l=strlen(temp->list);

if(l > 10 )

{

printf("\tPlease enter less characters in list name\n");

}

else

break;

}

do{

printf("\tenter price (1 to 999):");

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

if(temp->price > 1 && temp->price < 999)

{

break;

}

else

{

printf("\tPlease enter less price\n");

}

}while(1);

do{

printf("\tenter quantity(1 to 500):");

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

if(temp->quantity > 1 && temp->quantity < 500)

{

break;

}

else

{

printf("\tPlease enter less quantity\n");

}

}while(1);

do{

printf("\tenter receipt code(1 to 200):");

scanf("%d",&(temp->receipt\_code));

if(temp-> receipt\_code > 0 && temp-> receipt\_code < 200)

{

break;

}

else

{

printf("\tSORRY..not available Please enter receipt\_code between(1 to 200)\n");

}

}while(1);

printf("\n");

if(start==NULL)

{

start=temp;

temp->link=NULL;

}

else

{

p=start;

while(p->link!=NULL)

p=p->link;

p->link=temp;

temp->link=NULL;

}

}

return start;

}

void display (struct stationary \*start)

{

if(start==NULL)

{

printf("\n\*Order List is Empty\*\n\n");

return;

}

p=start;

printf("\n\tItem\tPrice\tquantity\treceipt no.\n");

while(p!=NULL)

{

printf("\n\t%s\t %d\t%d\t\t%d",p->list,p->price,p->quantity,p->receipt\_code);

p=p->link;

}

printf("\n");

}

struct stationary \*insert(struct stationary \*start)

{

struct stationary \*temp;

int pos;

temp=(struct stationary\*)malloc(sizeof(struct stationary)\*1);

printf("Enter location you want to insert:");

scanf("%d",&pos);

if(pos==1)

{

while(1)

{

printf("\tenter product name(Length 1 to 10):");

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

l=strlen(temp->list);

if(l > 10 )

{

printf("\tPlease enter less characters in list name\n");

}

else

break;

}

do{

printf("\tenter price(1 to 999):");

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

if(temp->price > 1 && temp->price < 999)

{

break;

}

else

{

printf("\tPlease enter less price\n");

}

}while(1);

do{

printf("\tenter quantity(1 to 500):");

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

if(temp->quantity > 1 && temp->quantity < 500)

{

break;

}

else

{

printf("\tPlease enter less quantity\n");

}

}while(1);

do{

printf("\tenter receipt code(1 to 200):");

scanf("%d",&(temp->receipt\_code));

if(temp-> receipt\_code > 0 && temp-> receipt\_code < 200)

{

break;

}

else

{

printf("\tSORRY..not available Please enter receipt\_code between(1 to 200)\n");

}

}while(1);

temp->link=start;

start=temp;

return start;

}

p=start;

for(i=1;i<pos-1 && p!=NULL;i++)

p=p->link;

if(p==NULL)

printf("\n\*less no of items\*\n\n");

else

{

while(1)

{

printf("\tenter product name(Length 1 to 10):");

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

l=strlen(temp->list);

if(l > 10 )

{

printf("\tPlease enter less characters in list name\n");

}

else

break;

}

do{

printf("\tenter price(1 to 999):");

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

if(temp->price > 1 && temp->price < 999)

{

break;

}

else

{

printf("\tPlease enter less price\n");

}

}while(1);

do{

printf("\tenter quantity(1 to 500):");

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

if(temp->quantity > 1 && temp->quantity < 500)

{

break;

}

else

{

printf("\tPlease enter less quantity\n");

}

}while(1);

do{

printf("\tenter receipt code(1 to 200):");

scanf("%d",&(temp->receipt\_code));

if(temp-> receipt\_code > 0 && temp-> receipt\_code < 200)

{

break;

}

else

{

printf("\tSORRY..not available Please enter receipt\_code between(1 to 200)\n");

}

}while(1);

temp->link=p->link;

p->link=temp;

}

return start;

}

struct stationary \*del(struct stationary \*start,char item[])

{

struct stationary \*temp;

if(strcmp(start->list,item)==0)

{

temp=start;

start=temp->link;

free(temp);

return start;

}

p=start;

while(p->link!=NULL)

{

if(strcmp(p->link->list,item)==0)

{

temp=p->link;

p->link=temp->link;

free(temp);

return start;

}

p=p->link;

}

}

struct stationary \*sort (struct stationary \*start)

{

struct stationary \*temp;

temp=(struct stationary\*)malloc(sizeof(struct stationary)\*1);

for(p=start;p->link!=NULL;p=p->link)

{

for(q=p->link;q!=NULL;q=q->link)

{

if(p->price > q->price)

{

temp->price=p->price;

p->price=q->price;

q->price=temp->price;

strcpy(temp->list,p->list);

strcpy(p->list,q->list);

strcpy(q->list,temp->list);

}

}

}

}

void \*search(struct stationary \*start,char item[])

{

if(strcmp(start->list,item)==0)

{

printf("\nITEM FOUND\n");

printf("\nItem\tPrice\tquantity\treceipt\_no.");

printf("\n%s\t %d\t %d\t %d\n",start->list,start->price,start->quantity,start->receipt\_code);

flag++;

}

p=start;

while(p->link!=NULL)

{

if(strcmp(p->link->list,item)==0)

{

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

printf("\nItem\tPrice\tquantity\treceipt\_no.");

printf("\n%s\t %d\t %d\t %d\n",p->link->list,p->link->price,p->link->quantity,p->link->receipt\_code);

}

p=p->link;

flag++;

}

if(flag==0)

printf("\n\nnot found\n\n");

}

struct stationary \*modify(struct stationary \*start,char item[])

{

char ans1,ans2;

int chw=0;

if(strcmp(start->list,item)==0)

{

printf("\nITEM FOUND\n");

printf("\nItem\tPrice\tquantity\treceipt\_no.");

printf("\n%s\t %d\t %d\t %d\n",start->list,start->price,start->quantity,start->receipt\_code);

do

{

printf("enter:\n 1) to modify list name\n 2) to modify list price:");

scanf("%d",&ch);

switch (ch)

{

case 1:

while(1)

{

printf("\tenter product name(Length 1 to 10):");

scanf("%s",(start->list));

l=strlen(start->list);

if(l > 10 )

{

printf("\tPlease enter less characters in list name\n");

}

else

{

break;

}

}

break;

case 2:

do{

printf("\tenter price(1 to 999):");

scanf("%d",&(start->price));

if(start->price > 1 && start->price < 999)

{

break;

}

else

{

printf("\tPlease enter less price\n");

}

}while(1);

break;

}

printf("press 1 to continue modifying:");

scanf("%d",&chw);

}while(chw==1);

flag++;

return start;

}

p=start;

while(p->link!=NULL)

{

if(strcmp(p->link->list,item)==0)

{

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

printf("\nItem\tPrice\tquantity\treceipt\_no.");

printf("\n%s\t %d\t %d\t %d\n",p->link->list,p->link->price,p->link->quantity,p->link->receipt\_code);

do

{

printf("enter:\n 1) to modify list name\n 2) to modify list price:");

scanf("%d",&ch);

switch (ch)

{

case 1:

while(1)

{

printf("\tenter product name(Length 1 to 10):");

scanf("%s",(p->link->list));

l=strlen(p->link->list);

if(l > 10 )

{

printf("\tPlease enter less characters in list name\n");

}

else

{

break;

}

}

break;

case 2:

do{

printf("\tenter price(1 to 999):");

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

if(p->link->price > 1 && p->link->price < 999)

{

break;

}

else

{

printf("\tPlease enter less price\n");

}

}while(1);

break;

}

printf("press 1 to continue modifying:");

scanf("%d",&chw);

}while(chw==1);

}

p=p->link;

flag++;

return start;

}

if(flag==0)

printf("\n\nnot found\n\n");

}

struct stationary\* getNewNode()

{

return (struct stationary\*)malloc(sizeof(struct stationary));

}

void create\_queue()

{

int q;

struct stationary \*temp;

printf("Enter no.of type of item:");

scanf("%d",&q);

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

{

temp=getNewNode();

while(1)

{

printf("\tenter product name(Length 1 to 10):");

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

l=strlen(temp->list);

if(l > 10 )

{

printf("\tPlease enter less characters in list name\n");

}

else

{

break;

}

}

do{

printf("\tenter price(1 to 999):");

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

if(temp->price > 1 && temp->price < 999)

{

break;

}

else

{

printf("\tPlease enter less price\n");

}

}while(1);

do{

printf("\tenter quantity(1 to 500):");

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

if(temp->quantity > 1 && temp->quantity < 500)

{

break;

}

else

{

printf("\tPlease enter less quantity\n");

}

}while(1);

do{

printf("\tenter receipt code(1 to 200):");

scanf("%d",&(temp->receipt\_code));

if(temp-> receipt\_code > 0 && temp-> receipt\_code < 200)

{

break;

}

else

{

printf("\tSORRY..not available Please enter receipt\_code between(1 to 200)\n");

}

}while(1);

printf("\n");

if (rear==NULL)

{

rear= temp;

front = temp;

rear->link=NULL;

continue;

}

rear->link=temp;

rear=temp;

}

}

void display\_queue()

{

if(front==NULL)

{

printf("\n No ITEM IN LIST\n");

}

else

{

p=front;

printf("\n\tLIST \tPRICE \tQUANTITY \tRECEIPT CODE \n\n");

while(p!=NULL)

{

printf("\t%s\t%d\t%d\t%d",p->list,p->price,p->quantity,p->receipt\_code);

printf("\n");

p=p->link;

}

}

}

void delete\_queue()

{

if(front==NULL)

{

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

}

else

{

temp=front;

front=front->link;

free(temp);

}

}

struct priceTree \*getTree(struct stationary \*start)

{

int arr[n];

struct stationary \*temp=start;

while(temp!=NULL)

{

arr[point++]=temp->price;

temp=temp->link;

}

sortTree(arr,n);

return BalancedBst(arr,0,n-1);

}

void sortTree(int arr[],int n)

{

int temp;

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

{

for(int j = 0; j < (n-1-i); j++)

{

if(arr[j] > arr[j+1]) //for decending order change "arr[j]<arr[j+1]"

{

temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

}

struct priceTree \*getnode(int x)

{

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

temp->price=x;

temp->left=temp->right=NULL;

return temp;

}

struct priceTree \*BalancedBst(int arr[],int s,int e)

{

if(s > e)

return NULL;

int mid=(s+e)/2;

struct priceTree \*root=getnode(arr[mid]);

root->left=BalancedBst(arr,s,mid-1);

root->right=BalancedBst(arr,mid+1,e);

return root;

}

void print2DUtil(struct priceTree \*root, int space)

{

if (root == NULL)

return;

space+=COUNT;

print2DUtil(root->right, space);

printf("\n");

for (int i = COUNT; i < space; i++)

printf(" ");

printf("%d\n\n", root->price);

print2DUtil(root->left, space);

}

void print2D(struct priceTree \*root)

{

print2DUtil(root, 0);

}

output:

Enter username:stationary Enter the password:\*\*\*\*\*

Welcome .....

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\* STATIONARY DISTRIBUTION SYSTEM \*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-1

Enter no of type of items:5

enter product name(Length 1 to 10):book

enter price (1 to 999):50

enter quantity(1 to 500):25

enter receipt code(1 to 200):1

enter product name(Length 1 to 10):pen

enter price (1 to 999):15

enter quantity(1 to 500):45

enter receipt code(1 to 200):2

enter product name(Length 1 to 10):file

enter price (1 to 999):30

enter quantity(1 to 500):55

enter receipt code(1 to 200):3

enter product name(Length 1 to 10):chart

enter price (1 to 999):10

enter quantity(1 to 500):20

enter receipt code(1 to 200):4

enter product name(Length 1 to 10):glue

enter price (1 to 999):21

enter quantity(1 to 500):88

enter receipt code(1 to 200):5

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-2

Item Price quantity receipt no.

book 50 25 1

pen 15 45 2

file 30 55 3

chart 10 20 4

glue 21 88 5

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-11

50

30

21

15

10

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-4

Enter item you want to delete:pen

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-2

Item Price quantity receipt no.

book 50 25 1

file 30 55 3

chart 10 20 4

glue 21 88 5

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-3

Enter location you want to insert:3

6

enter product name(Length 1 to 10):pen

enter price(1 to 999):15

enter quantity(1 to 500):55

enter receipt code(1 to 200):6

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-2

Item Price quantity receipt no.

book 50 25 1

file 30 55 3

pen 15 55 6

chart 10 20 4

glue 21 88 5

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-5

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-2

Item Price quantity receipt no.

chart 10 25 1

pen 15 55 3

glue 21 55 6

file 30 20 4

book 50 88 5

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-6

Enter item you want to search:glue

\*ITEM FOUND

Item Price quantity receipt\_no.

glue 21 55 6

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-7

Enter item you want to modify:pen

\*ITEM FOUND

Item Price quantity receipt\_no.

pen 15 55 3

enter:

1) to modify list name

2) to modify list price:2

enter price(1 to 999):20

press 1 to continue modifying:0

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-8

Enter no.of type of item:3

enter product name(Length 1 to 10):book

enter price(1 to 999):100

enter quantity(1 to 500):80

enter receipt code(1 to 200):7

enter product name(Length 1 to 10):eraser

enter price(1 to 999):8

enter quantity(1 to 500):60

enter receipt code(1 to 200):8

enter product name(Length 1 to 10):pencil

enter price(1 to 999):14

enter quantity(1 to 500):42

enter receipt code(1 to 200):9

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-9

LIST PRICE QUANTITY RECEIPT CODE

book 100 80 7

eraser 8 60 8

pencil 14 42 9

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-10

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-9

LIST PRICE QUANTITY RECEIPT CODE

eraser 8 60 8

pencil 14 42 9

1) to create

2) to display.

3) to insert.

4) to delete.

5) to sort by price.

6) to search.

7) to modify.

8) create using queue.

9) display using queue.

10) delete using queue.

11) Create tree of price.

12) exit(0).

Enter your choice-12