**SSN College of Engineering, Kalavakkam**

**Department of Computer Science and Engineering**

**III Semester**

**UCS 1302 Data Structure Lab**

**Academic Year: 2019-2020**

**Cruise Management**

R.Manoj kumar (185001092)

S.NachammaiDeviPooja (185001096)

S.Prathyush (185001112)

V.PraveenKumar (185001113)

**Abstract:**

The Cruise Management application has been built using data structures such as graphs and linked list. Using this application, the user can perform operations such as booking rooms, check for confirmation of booking, see the set path of ship ,finding shortest path between two points ,display payroll and deletion of rooms.

**Methodology:**

The diffenet functions implemented using this appliction are :

* room booking
* finding shortest path between pickup and drop point
* payment method
* payroll display
* deletion/checkout of rooms
* check availability of rooms
* check booking status

**CONCEPTS USED:**

* **Linked list :**

This data structure is used to handle different passengers in cruise.

* **Shortest path algorithm :**

This data structure is used to find shortest path between two points.

* **Bfs & Dfs:**

These traversals are used to find distance between locations.

* **Queue :**

This data structure is used to add passengers according to the time of entry.

**FUNCTIONS :**

**graph.h:**

* **void dijkstra(int G[MAX][MAX],int n,int startnode) :**

This function calculates shortest path between two nodes and prints them. The calculation is done when when distance between the nodes is infinity.

* **void insert(int adj[][5],int src,int dest):**

This function is used to insert nodes in the graph to find the shortest path.the nodes are considered as places .

* **void display(int adj[][5],int n):**

This function is used to print the graph sent to this function.

* **int visited1(int ele,int ar[],int n):**

This function is used to mark visited nodes once they are visited.

* **int dfs(int start,int adj[][5],int ar[],int size):**

This function is used to do depth wise traversal of graph.

* i**nt bfs(int start,int adj[][5],int ar[],int size):**

Thisfunction is used to breth wise traversal of graph.

**Passengerlist.h :**

* **List CreateEmptyList():**

This function is used to create empty list and return it.

* **int Find(List L,char X[])**:

This function is used to find the passenger is already present or not.This function returns 1 if passenger is present.

* **void addend(List L, details d):**

This function is used to add passengers in list .

* **void Delete(List L, char X[]):**

This function is used to delete a passenger from list .

* **void disp(List h):**

This function is used to display list of passengers.

* **void checkavailabilty(char a[]):**

This function is used to check availability of rooms present in the cruise.

**Ports.h:**

* **void init(int n):**

This function is used to add initial graph nodes.

* **void addEdge(int src, int dest, char ar[],int dist):**

This function is used to add connections between nodes .

* **void printGraph(int n):**

This function is used to print graph.

* **void ListtoMatrix(int G[][5]):**

This function is used to find adjacency matrix.

* **void stationInitialise(int G[][5]):**

This function is used to initialise each ports the route of cruise.

**Stack.h :**

* **void push(int value):**

This function pushes the element into stack.

* **int Pop():**

This function returns the top element in stack

* **int peek():**

This function returns top element if stack.

* **int isEmpty():**

This function return a stack is empty or not

* **void stackdisplay():**

This function is used to display the stack elements.

**SOURCE CODE**

STACK.H:

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

typedef struct s

{

int x;

struct s \*next;

}snode;

snode \*top=NULL;

void push(int value)

{

snode \*New;

New=malloc(sizeof(snode));

New->x=value;

if(top==NULL)

New->next=NULL;

else

New->next=top;

top=New;

}

int Pop()

{

int ele=-1;

if(top==NULL)

printf("Stack Empty!!!\n");

else

{

snode \*temp=top;

top=temp->next;

ele=temp->x;

free(temp);

}

return ele;

}

int peek()

{

if(top==NULL)

printf("Stack Empty\n");

else

return top->x;

return -1;

}

int isEmpty()

{

if(top==NULL)

return 1;

else

return 0;

}

void stackdisplay()

{

if(top==NULL)

printf("Stack Empty");

else

{

snode \*temp=top;

while(temp->next!=NULL)

{

printf("%d",temp->x);

temp=temp->next;

}

printf("%d",temp->x);

}

}

PORTS.H:

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct node

{

char dest[100];

int vertex;

int dist;

struct node\* link;

};

struct node\* graph[20];

char p[][20] = {"Chennai","Mumbai","Andhaman","Lakshwadeep","Sri Lanka"};

void init(int n)

{

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

{

graph[i]=malloc(sizeof(struct node));

graph[i]->vertex=-1;

graph[i]->dist=0;

strcpy(graph[i]->dest,p[i]);

graph[i]->link=NULL;

}

return ;

}

void addEdge(int src, int dest, char ar[],int dist)

{

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

New->vertex=dest;

New->link=NULL;

strcpy(New->dest,ar);

New->dist=dist;

struct node\* temp=graph[src];

while(temp->link!=NULL)

{

temp=temp->link;

}

temp->link=New;

}

void printGraph(int n)

{

int v;

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

{

printf("%s -> ",graph[v]->dest);

struct node\* temp = graph[v]->link;

while (temp!=NULL)

{

printf("%s(%d) ", temp->dest,temp->dist);

temp = temp->link;

}

printf("\n");

}

}

void ListtoMatrix(int G[][5])

{

for (int v = 0; v < 5; ++v)

{

struct node\* temp = graph[v]->link;

while (temp!=NULL)

{

G[v][temp->vertex]=temp->dist;

temp = temp->link;

}

}

}

void stationInitialise(int G[][5])

{

init(5);

/\*

0-Chennai

1-Villupuram

2-Pondicherry

3-Madurai

4-Coimbatore

\*/

addEdge(0,4,p[4],1680);

addEdge(4,0,p[0],1680);

addEdge(0,1,p[1],5130);

addEdge(1,0,p[0],5130);

addEdge(0,3,p[3],1650);

addEdge(3,0,p[0],1650);

addEdge(4,1,p[1],3370);

addEdge(1,4,p[4],3370);

addEdge(4,2,p[2],2940);

addEdge(2,4,p[4],2940);

addEdge(4,3,p[3],910);

addEdge(3,4,p[4],910);

addEdge(3,2,p[2],3000);

addEdge(2,3,p[3],3000);

addEdge(2,1,p[1],2360);

addEdge(1,2,p[2],2360);

//printGraph(5);

for(int i=0;i<5;i++)

for(int j=0;j<5;j++)

G[i][j]=0;

}

GRAPH.H:

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include "stack.h"

#include "ports.h"

#define Infinity 9999

#define MAX 5

void dijkstra(int G[MAX][MAX],int n,int startnode)

{

int cost[MAX][MAX],distance[MAX],pred[MAX];

int visited[MAX],count,mindistance,nextnode,i,j;

//pred[] stores the predecessor of each node

//count gives the number of nodes seen so far

//create the cost matrix

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

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

if(G[i][j]==0)

cost[i][j]=Infinity;

else

cost[i][j]=G[i][j];

//initialize pred[],distance[] and visited[]

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

{

distance[i]=cost[startnode][i];

pred[i]=startnode;

visited[i]=0;

}

distance[startnode]=0;

visited[startnode]=1;

count=1;

while(count<n)

{

mindistance=Infinity;

//nextnode gives the node at minimum distance

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

if(distance[i]<mindistance&&!visited[i])

{

mindistance=distance[i];

nextnode=i;

}

//check if a better path exists through nextnode

visited[nextnode]=1;

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

if(!visited[i])

if(mindistance+cost[nextnode][i]<distance[i])

{

distance[i]=mindistance+cost[nextnode][i];

pred[i]=nextnode;

}

count++;

}

char \*out[10];

int k;

//print the path and distance of each node

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

if(i!=startnode)

{

printf("\n\nDistance for Station %s = %d",p[i],distance[i]);

printf("\nPath => %s",p[i]);

j=i;

k=0;

do

{

j=pred[j];

strcpy(out[k++],p[j]);

//printf(" <-%s",p[j]);

}while(j!=startnode);

do{

printf(" <--%s",out[k--]);

}while(k>=0);

}

}

void initial(int adj[][5],int n)

{

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

{

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

{

adj[i][j]=0;

}

}

}

void insert(int adj[][5],int src,int dest)

{

adj[src][dest]=1;

}

void display(int adj[][5],int n)

{

printf("-\t");

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

{

printf("%d\t",i);

}

printf("\n \t");

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

{

printf("-------");

}

printf("\n");

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

{

printf("%d|\t",i);

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

{

printf("%d\t",adj[i][j]);

}

printf("\n");

}

}

int visited1(int ele,int ar[],int n)

{

int flag=0;

if (n==0)

{

return 0;

}

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

{

if (ar[i]==ele)

{

flag=1;

break;

}

}

return flag;

}

int dfs(int start,int adj[][5],int ar[],int size)

{

int i=0;

push(start);

while(!isEmpty())

{

i=Pop();

if(!visited1(i,ar,size))

{

ar[size]=i;

size++;

}

for(int j=5-1;j>=0;j--)

{

if(adj[i][j]!=0)

{

if(!visited1(j,ar,size))

push(j);

}

}

}

return size;

}

PASSENGERLIST.H:

#include<stdlib.h>

#include<stdio.h>

#include<string.h>

typedef struct Node \*PtrToNode;

typedef PtrToNode List;

typedef PtrToNode Position;

typedef struct dett

{

char name[100];

char gender;

int age;

char proof[50];

char Class[50];

char src[50];

char dest[50];

}details;

struct Node

{

details det;

Position Next;

Position Prev;

};

List CreateEmptyList()

{

List t,h;

h=(List)malloc(sizeof(struct Node));

t=(List)malloc(sizeof(struct Node));

h->Prev=NULL;

h->Next=t;

t->Prev=h;

t->Next=NULL;

return h;

}

int IsEmpty( List L )

{

return L->Next == NULL;

}

int Find(List L,char X[])

{

int i=1;

Position P;

P = L->Next;

while( (P->Next) != NULL && strcmp(P->det.name,X)!=0 )

{

P = P->Next;

i++;

}

if(P->Next == NULL){

printf("No name found.\n");

return -1;

}

else

{

printf(" Name is in position %d\n",i);

printf("Name:%s\n",P->det.name);

printf("Gender:%c\n",P->det.gender);

printf("Age:%d\n",P->det.age);

printf("Proof:%s\n",P->det.proof);

printf("Class:%s\n",P->det.Class);

}

return 1;

}

void addend(List L, details d)

{

Position Temp,P;

int i;

P=L;

while(P->Next!=NULL)

{

P=P->Next;

}

P=P->Prev;

Temp =(List) malloc( sizeof( struct Node ) );

Temp->det=d;

Temp->Next = P->Next;

Temp->Prev = P;

Temp->Next->Prev=Temp;

P->Next=Temp;

}

void Delete(List L, char X[])

{

if(L==NULL)

{

printf("\n%s is not present in the list", X);

return;

}

Position P,temp;

P = L->Next;

while( P!= NULL && (strcmp(P->det.name,X)!=0) )

{

P = P->Next;

}

if(P!=NULL)

{

temp = P;

temp -> Next -> Prev = temp -> Prev;

temp -> Prev -> Next = temp -> Next;

free(temp);

}

else

printf("\n%s is not present in the list\n", X);

}

void disp(List h)

{

List temp;

temp=(List)malloc(sizeof(struct Node));

temp=h->Next;

printf("All records.\n");

while(temp->Next!=NULL)

{

printf("\n%s",temp->det.name);

temp=temp->Next;

}

printf("\n");

}

void checkavailabilty(char a[])

{

FILE \*fptr;

fptr=fopen("today.txt","r");

char c[10];

int i,j=1,k=1,co=0,co1=0;

for(i=0;i<10;i++)

{

fscanf(fptr,"%s",c);

{

if(c[1]<65)

{

a[j]=c[0];

j++;

co++;

}

}

}

fclose(fptr);

while(j<10)

{

a[j]='\0';

j++;

}

if(co==0) a[0]='n';

else a[0]='y';

return;

}

CRUISE.C:

#include<stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include"passengerlist.h"

#include "graph.h"

typedef struct

{

char source;

char destination;

}route;

typedef struct mynode

{

char room[25];

struct mynode \*next;

}node;

void displaystations()

{

printf("Stations : \n");

for(int i=0;i<5;i++)

{

printf("%d ) %s\n",i,p[i]);

}

}

int findindex(char str[])

{

for(int i=0;i<5;i++)

{

if(!strcmp(p[i],str))

return i;

}

return -1;

}

void Status(List L)

{

char name[100];

printf("Enter the name to know status:");

scanf("%s",name);

int i=-1;

printf("\tDetails");

i=Find(L,name);

if(i!=-1)

{

printf("Status:Your tickets are confirmed.\n");

}

else

printf("Status: You have no confirmed booking\n");

}

void Chart()

{

printf("Reservation of rooms.\n");

int i=1;

int x=0;

FILE \*fptr;

char b1[20],a1[20];

fptr=fopen("today.txt","r");

for(i=0;i<10;i++)

{

x=0;

fscanf(fptr,"%s",b1);

while(b1[x]!='\0'){

a1[x-1]=b1[x];

x++;

}

a1[x-1]='\0';

while(a1[0]!='\0'){

printf("\nName : %s Room : %c",a1,b1[0]);

break;

}

}

fclose(fptr);

}

void Routes(int G[][5])

{

int fs[5],size;

int ch=1;

char Start[50];

int start;

while(ch!=0)

{

printf("\n\n\t\t1.Cruise Set Path\n\t\t2.Possiible Routes from selected Port\n\t\t0.Exit\n\t\tEnter Choice : ");

scanf(" %d",&ch);

switch(ch)

{

case 1 :

displaystations();

printf("\nEnter Boarding Station : ");

scanf("%d",&start);

size=0;

size=dfs(start,G,fs,size);

printf("DFS : \n");

for(int i=0;i<5;i++)

printf("-->%s --",p[fs[i]]);

printf("\n");

break;

case 2 :

printf("\nAll available Boarding Port : ");

displaystations();

printf("\nEnter Boarding Port : ");

scanf("%d",&start);

dijkstra(G,5,start);

break;

case 0 : break;

default : printf("Invalid Choice \n");

}

}

}

node\* insert1()

{

FILE \*fptr;

node \*p,\*head,\*temp;

int i,j=0;

char ch[20];

fptr=fopen("today.txt","r");

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

head=p;

for(i=0;i<30;i++)

{

fscanf(fptr,"%s",ch);

strcpy(p->room,ch);

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

p->next=temp;

p=p->next;

}

fclose(fptr);

return head;

}

node\* cancel\_1(node \*a,char b[],int c)

{

int i;

node \*temp;

char ch[20];

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

{

temp=a;

while(temp->next!=NULL)

{

if(temp->room[0]==b[i])

{

ch[0]=temp->room[0];

strcpy(temp->room,ch);

break;

}

else temp=temp->next;

}

}

return a;

}

void cancel(char a[])

{

FILE \*fptr;

int i,j=0,n,t=0,l,t1;

char ch[25],c1[25],se,li[25],c[25];

node \*p;

fptr=fopen("today.txt","r");

for(i=48;i<58;i++)

{

ch[0]=i;

for(t1=1;t1<25;t1++) ch[t1]='\0';

fscanf(fptr,"%s",c);

strcat(ch,a);

if((strcmp(c,ch))==0)

{

c1[j]=c[0];

j++;

}

}

fclose(fptr);

if(c1[0]=='\0') printf("\nYou don't have any booking on this date\n");

else

{

printf("\nrooms booked by you\n");

for(i=0;i<j;i++) printf("%c ",c1[i]);

printf("\nkindly select no of rooms you want to cancel\n");

j=1;

do

{

scanf("%d",&n);

if(n<strlen(c1)) j=0;

else printf("\nkindly select correct no of rooms\n");

}while(j);

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

{

printf("room-%d:",j+1);

scanf("\n%c",&se);

for(l=0;l<strlen(c1);l++)

{

if(c1[l]==se) break;

}

if(l<strlen(c1))

{

li[t]=se;

t++;

}

else

{

printf("kindly select rooms booked by you\n");

j--;

}

}

p=insert1();

p=cancel\_1(p,li,t);

update(p,"today.txt");

}

}

node\* book\_1(node\* a,char b[],char c[],int d)

{

int i;

node \*temp;

FILE \*fptr;

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

{

temp=a;

while(temp->next!=NULL)

{

if(temp->room[0]==b[i])

{

strcat(temp->room,c);

break;

}

else temp=temp->next;

}

}

return a;

}

void payment(int a,int n)

{

int age,fare,tax,i;

char ca[20],cv[20],ot[20],ch[20];

if(a==1) fare = 50000;

else if(a==2) fare = 20000;

else fare = 10000;

tax=(0.2\*fare);

printf("\nCruise Fare Summary\n\n");

printf("\nPer Family:\n");

printf("Cruise Fare\t\t%d\n",fare);

printf("Taxes \t\t%d\n",tax);

printf("Total \t\t%d\n",fare+tax);

printf("\n\nTotal Cruise Fare\t\t%d.00\n",((fare+tax)\*n));

printf("\nPAYMENT GATEWAY\n");

printf("\nEnter 1-Debit card/2-Credit card/3-Netbanking/4-UPI\n");

scanf("%d",&i);

printf("\n");

if(i==1)

{

printf("\nEnter card number:");

scanf("\n%s",ca);

printf("\nEnter cvv number:");

scanf("\n%s",cv);

printf("\nRedirecting to payment portal\n");

printf("\nEnter OTP:");

scanf("\n%s",ot);

printf("\nPayment Processing\n");

}

else if(i==2)

{

printf("\nEnter card number:");

scanf("\n%s",ca);

printf("\nEnter cvv number:");

scanf("\n%s",cv);

printf("\nRedirecting to payment portal\n");

printf("\nEnter OTP:");

scanf("\n%s",ot);

printf("\nPayment Processing\n");

}

else if(i==3)

{

printf("\nEnter username:");

scanf("\n%s",ca);

printf("\nEnter password:");

scanf("\n%s",cv);

printf("\nRedirecting to payment portal\n");

printf("\nEnter OTP:");

scanf("\n%s",ot);

printf("\nPayment Processing\n");

}

else

{

printf("\nEnter UPI password:");

scanf("\n%s",ca);

printf("\nEnter UPI pin:");

scanf("\n%s",cv);

printf("\nPayment Processing\n");

}

printf("\nPayment Successful\n");

printf("\nTickets have been booked successfully\n\n");

printf("Thankyou for choosing us\n");

printf("Wish You A Safe And Happy Journey\n");

return;

}

void Cancel1(List L)

{

printf("\nEnter the name of Person on ticket to cancel");

char name[100];

scanf("%s",name);

Delete(L,name);

cancel(name);

printf("\nCancelled ticket successfully.\n");

}

void update(node \*a,char c[20])

{

FILE \*fptr;

node \*temp;

temp=a;

fptr=fopen(c,"w");

while(temp->next!=NULL)

{

fprintf(fptr,"%s\n",temp->room);

temp=temp->next;

}

fclose(fptr);

}

void Book(List L)

{

char a[30];

checkavailabilty(a);

details det;

printf("\nAvailable rooms\n");

for(int l=1;l<strlen(a);l++)

{

printf("%c ",a[l]);

}

node \*p,\*p1;

int j,l,i=0,n;

char li[20],c1[20],se;

printf("\nKindly select no of Rooms\n");

j=1;

do

{

scanf("%d",&n);

if(n<strlen(a)) j=0;

else printf("\nKindly select correct no of Rooms\n");

}while(j);

printf("\nKindly select your room\n");

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

{

printf("Room-%d:",j+1);

scanf("\n%c",&se);

for(l=1;l<strlen(a);l++)

{

if(a[l]==se) break;

}

if(l<strlen(a))

{

li[i]=se;

i++;

}

else

{

printf("kindly select available rooms\n");

j--;

}

}

printf("\nEnter details of passengers\n\n");

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

{

printf("\nPassenger-%d Details:",i+i);

printf("\nEnter name:");

scanf("%s",det.name);

printf("Enter gender(M/F/T):");

scanf(" %c",&det.gender);

printf("Enter age:");

scanf("%d",&det.age);

printf("Enter proof of identity:");

scanf("%s",det.proof);

printf("Enter class(First/Second/Third):");

scanf("%s",det.Class);

printf("Enter the boarding Port:");

scanf("%s",det.src);

printf("Enter the destination:");

scanf("%s",det.dest);

addend(L,det);

}

p=insert1();

p=book\_1(p,li,det.name,i);

update(p,"today.txt");

if(det.Class[0]=='f'||det.Class[0]=='F')payment(1,n);

else if(det.Class[0]=='s'||det.Class[0]=='S')payment(2,n);

else payment(3,n);

return;

}

int main()

{

details d[100];

route r[100];

List L=CreateEmptyList();

int ch;

int Graph[5][5];

stationInitialise(Graph);

while(1)

{

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

printf("\n\t\t1.Book\n\t\t2.Booking Status\n\t\t3.Cancel Booking\n\t\t4.Routes available\n\t\t5.Room Availability");

printf("\n\t\t0.Exit\n\t\tEnter Choice : ");

scanf("%d",&ch);

switch(ch)

{

case 1 :Book(L);break;

case 2 : Status(L);

break;

case 3 : Cancel1(L);

printf("Refund process is initiated\n");

printf("You will receive your refund within 7-working days\n");

break;

case 4 :Routes(Graph);

break;

case 5 : Chart();

break;

case 0 : exit(0);

default : printf("Invalid Choice ");

}

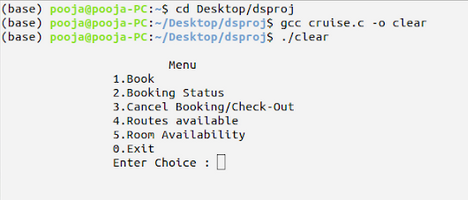
}

return 0;

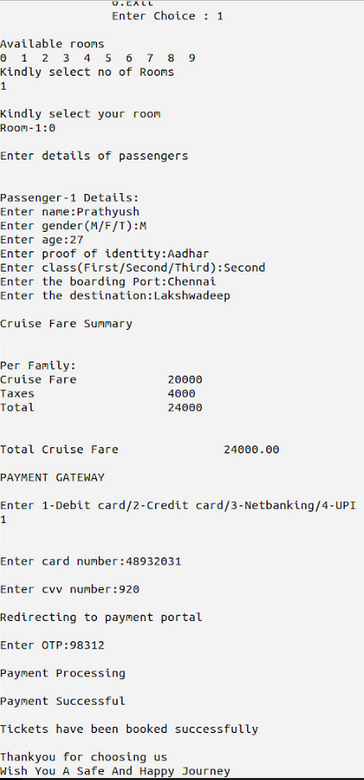
}

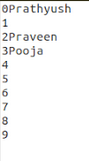
**OUTPUT SCREENSHOTS :**

* initial promt message when we start executing the main cruise.c file is shown below.



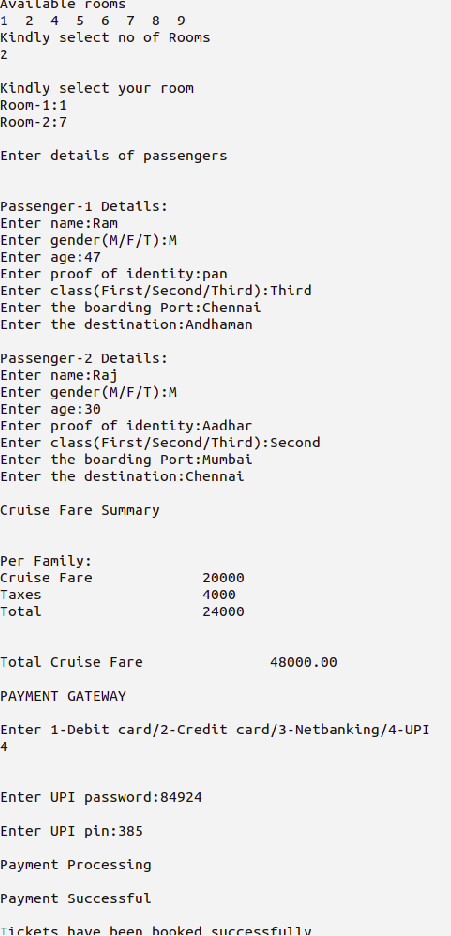
* For Booking we select choice 1. We get the details for the booking as shown below.



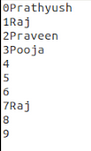


* On addition of similar details we see a change in the text file today.txt for each change

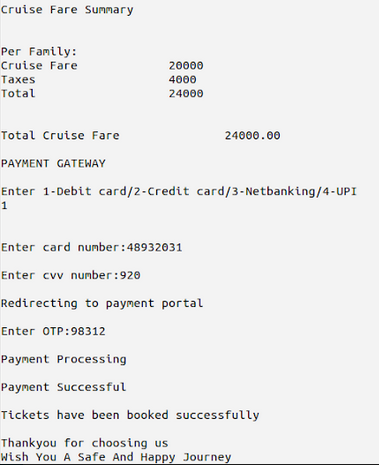
We can also add two rooms in a booking,which is shown below



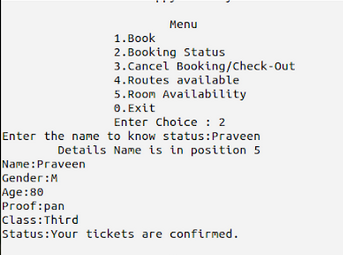
The file after modification is :-



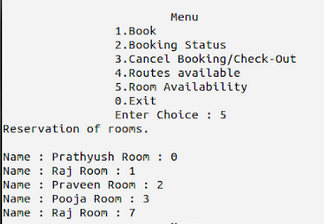
* The Payroll is displayed here after booking.



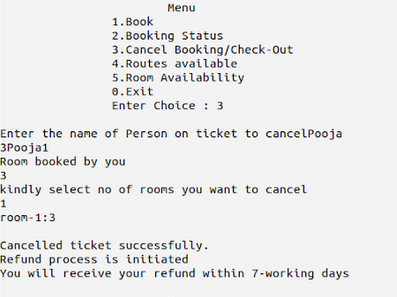
* The customer can check there booking status by selecting choice as 2. The Booking Status is shown below.



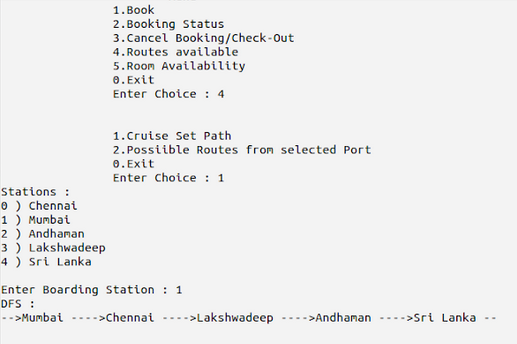
* The Customer can check room availability before booking by selecting choice as 5.

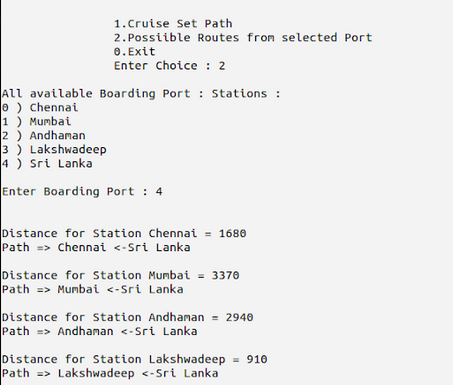


* To cancel tickets we choose choice as 3,and enter the person to delete.



* To check for a reqiured route from desired destination select choice as 4. In that we have two choices Cruise Set Path and Possible Routes.





**Conclusion:**

Thus cruise management application is created and tested successfully . The use of data structures has enhanced the way in which the operations are done and has provided additional functionality to the application.