SOURCE CODE:

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<termios.h>

#include<string.h>

#include<time.h>

#include<ctype.h>


char* cities[]={"CHENNAI", "BANGALORE", "PUNE", "DELHI", "KOCHI", "MUMBAI", "HYDERABAD",
"SINGAPORE", "LONDON", "DUBAI"};

char *fnames[]={"123.bin", "456.bin", "789.bin"};


struct location

{

   char street[40];

   char city[25];

   char pincode[10];

   char state[25];

};


struct date

{

   int day;

   int month;

   int year;

};

FILE *f;
```

```
struct user_details
{
    char username[40];
    char password[40];
    char name[40];
    struct location address;
    char nationality[40];
    char mobile[12];
    char email[45];
    struct date dob;
    int age;
    char gender;
};

// Identifiers used for admin login

#define USERNAME     "admin"
#define PASSWORD    "admin20"

/* Function to implement getch() due to absence of <conio.h> in gcc compliler */

char getch()
{
    char buf=0;
    struct termios old={0};
    fflush(stdout);

    if(tcgetattr(0, &old)<0)
        perror("tcsetattr()");
    old.c_lflag&=~ICANON;
        old.c_lflag&=~ECHO;
        old.c_cc[VMIN]=1;
        old.c_cc[VTIME]=0;

    if(tcsetattr(0, TCSANOW, &old)<0)
        perror("tcsetattr ICANON");

    if(read(0,&buf,1)<0)
        perror("read()");

    old.c_lflag|=ICANON;
       old.c_lflag|=ECHO;

    if(tcsetattr(0, TCSADRAIN, &old)<0)
        perror ("tcsetattr ~ICANON");

    return buf;
}

// Function to display the introductory opening page

void intro()
{
    system("clear");

    printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t\t\t\t\t\t\t  MINI PROJECT\n");
    printf("\n\t\t\t\t\t\t\t\t\t\t  Airplane Reservation System");
```

```
   printf("\n\n\n\t\t\t\t\t\t\t\t\t\tMADE BY: Lakshmi Priya 185001083\n \t\t\t\t\t\t\t\t\t\t Nivedhitha
D  185001104\n\t\t\t\t\t\t\t\t\t\t Sakthi Sairaj 185001134\n\n");

     printf("\t\t\t\t\t\t\t\t\t\tBRANCH : CSE B\n");

   printf("\n\t\t\t\t\t\t\t\t\tCOLLEGE: SSN College of Engineering\n");


   getch();

}


void ticket_enquiry()

{

   system("clear");


   char ch;


   printf("\n\n\n\tBOOKED TICKET ENQUIRY");

   printf("\n\n\tTICKET OPTIONS");

   printf("\n\n\t\t1. View Particular Flight Ticket");

   printf("\n\n\t\t2. View All Flight Tickets");

   printf("\n\n\tBILL OPTIONS");

   printf("\n\n\t\t3. View A Particular Bill");

   printf("\n\n\t\t4. View All Bills");

   printf("\n\n\tEXIT OPTION");

   printf("\n\n\t\t5. BACK TO USER MENU");


   printf("\n\n\n\n\tEnter choice: ");

   scanf(" %c", &ch);


   switch(ch)
```

```
    {
      case '1':   system("clear");
              getch();
              ticket_enquiry();
              break;

      case '2':   system("clear");
              getch();
              ticket_enquiry();
              break;

      case '3':   system("clear");
              getch();
              ticket_enquiry();
              break;

      case '4':   system("clear");
              getch();
              ticket_enquiry();
              break;

      case '5':   system("clear");
              break;

      default:   ticket_enquiry();
    }
  }
```

```
long int read_count_users(struct user_details all_users[])
{
   long int count=0;
   FILE *fptr=fopen("user.bin", "rb");

   fseek(fptr, 0, SEEK_END);
   count=(ftell(fptr))/sizeof(struct user_details);

   fseek(fptr, 0, SEEK_SET);

   for(int i=0; i<count; i++)
      fread(&all_users[i], sizeof(struct user_details), 1, fptr);

   fclose(fptr);

   return count;
}

int search_users(int n, struct user_details all_users[])
{
   char search[40];

   printf("\n\tCURRENT Username: ");
   scanf("%s",search);

   for (int i=0; i<n; i++)
      if(strcmp(all_users[i].username, search)==0)
            return i;
```

```
    return -1;
}

void user_det_update()
{
    struct user_details all_users[10];
    int count=0, index=-1;

    count=read_count_users(all_users);

    printf("\n\n\n\tUPDATE USER ACCOUNT PROFILE\n");

    index=search_users(count, all_users);

    FILE *fptr=fopen("user.bin", "wb");

    if(index!=-1)
    {
        char ch;

        do
        {    system("clear");

            printf("\n\n\n\tUPDATE USER ACCOUNT PROFILE\n");

            printf("\n\n\t0. Username");
            printf("\n\n\t1. Password");
            printf("\n\n\t2. Name");
            printf("\n\n\t3. Address");
            printf("\n\n\t4. Nationality");

            printf("\n\n\t5. Mobile Number");

            printf("\n\n\t6. Email ID");

            printf("\n\n\t7. Date Of Birth");

            printf("\n\n\t8. Age");

            printf("\n\n\t9. Gender");

            printf("\n\n\te. BACK TO USER MENU");


            printf("\n\n\n\n\tEnter choice: ");

            scanf(" %c", &ch);


            switch(ch)

            {

               case '0':    system("clear");

                      printf("\n\n\n\tUPDATE USER ACCOUNT PROFILE\n");

                      printf("\n\n\tEnter the NEW Details\n\n");

                      printf("\tUSERNAME: ");

                      scanf(" %[^\n]", all_users[index].username);

                      break;


               case '1':    system("clear");

                      printf("\n\n\n\tUPDATE USER ACCOUNT PROFILE\n");

                      printf("\n\n\tEnter the NEW Details\n\n");

                      printf("\tPASSWORD: ");

                      scanf(" %[^\n]", all_users[index].password);                    break;


               case '2':    system("clear");

                      printf("\n\n\n\tUPDATE USER ACCOUNT PROFILE\n");

                      printf("\n\n\tEnter the NEW Details\n\n");
```

```
        printf("\n\tName: ");
        scanf(" %[^\n]", all_users[index].name);
        break;

case '3':    system("clear");
        printf("\n\n\n\tUPDATE USER ACCOUNT PROFILE\n");
        printf("\n\n\tEnter the NEW Details\n\n");
        printf("\n\tAddress:\n");
        printf("\tStreet: ");
        scanf(" %[^\n]", all_users[index].address.street);

        printf("\tCity: ");
        scanf(" %[^\n]", all_users[index].address.city);

        printf("\tPincode: ");
        scanf(" %[^\n]", all_users[index].address.pincode);

        printf("\tState: ");
        scanf(" %[^\n]", all_users[index].address.state);
        break;

case '4':    system("clear");
        printf("\n\n\n\tUPDATE USER ACCOUNT PROFILE\n");
        printf("\n\n\tEnter the NEW Details\n\n");              printf("\n\tNationality: ");
        scanf(" %[^\n]", all_users[index].nationality);
        break;

case '5':    system("clear");
        printf("\n\n\n\tUPDATE USER ACCOUNT PROFILE\n");
```

```
                printf("\n\n\tEnter the NEW Details\n\n");
                printf("\tMobile: ");
                scanf(" %[^\n]", all_users[index].mobile);
                break;

            case '6':    system("clear");
                printf("\n\n\n\tUPDATE USER ACCOUNT PROFILE\n");
                printf("\n\n\tEnter the NEW Details\n\n");
                printf("\tEmail ID: ");
                scanf(" %[^\n]", all_users[index].email);
                break;

            case '7':    system("clear");
                printf("\n\n\n\tUPDATE USER ACCOUNT PROFILE\n");
                printf("\n\n\tEnter the NEW Details\n\n");
                printf("\n\tEnter DATE in dd mm yyyy FORMAT\n");
                printf("\tDate of birth: ");
                scanf("%d %d %d", &all_users[index].dob.day, &all_users[index].dob.month,
&all_users[index].dob.year);
                break;

            case '8':    system("clear");
                printf("\n\n\n\tUPDATE USER ACCOUNT PROFILE\n");
                printf("\n\n\tEnter the NEW Details\n\n");
                printf("\n\tAge: ");
          scanf("%d", &all_users[index].age);
                break;

            case '9':    system("clear");
```

```
                printf("\n\n\n\tUPDATE USER ACCOUNT PROFILE\n");
                printf("\n\n\tEnter the NEW Details\n\n");
                printf("\n\tF: Female M: Male T:Transgender O:Other\n");
                printf("\tGender: ");
                scanf(" %c", &all_users[index].gender);
                break;

            case 'e':    printf("\n\n\n\tUPDATED USER ACCOUNT PROFILE SUCCESSFULLY...!!!\n");
                getch();
                break;

            default:    system("clear");
                printf("\n\n\n\tUPDATE USER ACCOUNT PROFILE\n");
                printf("\n\nINVALID CHOICE\n");
                getch();

        }
    }while(ch!='e');

    fseek(fptr, 0, SEEK_SET);
    fwrite(&all_users, sizeof(struct user_details), count, fptr);

}

else
{
    printf("\n\n\n\tUSER NOT FOUND...!!!");
    printf("\n\n\tINVALID USERNAME/DOESN'T EXIST...!!!\n");
}
```

```
    fclose(fptr);

}

void user_det_delete()
{
    struct user_details all_users[10];
    int count=0, index=-1;

    printf("\n\n\n\tDELETE USER PROFILE\n\n");

    count=read_count_users(all_users);

    index=search_users(count, all_users);

    FILE *fptr=fopen("user.bin", "wb");

    if(index!=-1)
    {
        for(int i=index; i<(count-1); i++)
            all_users[i]=all_users[i+1];

        fseek(fptr, 0, SEEK_SET);

        for(int i=0; i<(count-2); i++)
            fwrite(&all_users[i], sizeof(struct user_details), 1, fptr);

        printf("\n\n\n\tDELETEED USER ACCOUNT PROFILE SUCCESSFULLY...!!!\n");
    }

    else
    {
        printf("\n\n\n\tUSER NOT FOUND...!!!");
        printf("\n\n\tINVALID USERNAME/DOESN'T EXIST...!!!\n");
    }

    fclose(fptr);

    getch();
}

void user_det_view_particular()
{
    struct user_details all_users[10];
    int count=0, index=-1;

    printf("\n\n\n\tVIEW A PARTICULAR USER ACCOUNT PROFILE\n\n");

    count=read_count_users(all_users);

    index=search_users(count, all_users);

    if(index!=-1)
    {
        system("clear");

        printf("\n\n\n\tVIEW A PARTICULAR USER ACCOUNT PROFILE\n\n");
        printf("\tUSERNAME: %s\n", all_users[index].username);


        printf("\tPASSWORD: %s\n", all_users[index].password);


        printf("\n\tName: %s\n", all_users[index].name);


        printf("\n\tAddress:\n");
        printf("\t%s\n\t%s - %s\n\t%s\n", all_users[index].address.street, all_users[index].address.city,
all_users[index].address.pincode, all_users[index].address.state);


        printf("\n\tNationality: %s\n", all_users[index].nationality);


        printf("\tMobile: %s\n", all_users[index].mobile);


        printf("\tEmail ID: %s\n", all_users[index].email);


        printf("\tDate of birth: %d-%d-%d\n", all_users[index].dob.day, all_users[index].dob.month,
all_users[index].dob.year);


        printf("\n\tAge: %d\n", all_users[index].age);


        printf("\n\tF: Female M: Male T:Transgender O:Other\n");
        printf("\tGender: %c\n", all_users[index].gender);
    }


    else
    {
        printf("\n\n\n\tUSER NOT FOUND...!!!");
        printf("\n\n\tINVALID USERNAME/DOESN'T EXIST...!!!\n");
    }

    getch();

}



void user_det_view_all()
{
    struct user_details all_users[10];
    int count=-1, i=0;
    FILE *fptr= fopen("user.bin", "rb");

    printf("\n\n\n\tVIEW All USER ACCOUNT PROFILE");

    count=read_count_users(all_users);

    if(count==0)
        count=-1;

    while((i<count))
    {
        printf("\n\n\tACCOUNT PROFILE DETAILS\n\n");

        printf("\tUSERNAME: %s\n", all_users[i].username);

        printf("\tPASSWORD: %s\n", all_users[i].password);
        printf("\n\tName: %s\n", all_users[i].name);


        printf("\n\tAddress:\n");
        printf("\t%s\n\t%s - %s\n\t%s\n", all_users[i].address.street, all_users[i].address.city,
all_users[i].address.pincode, all_users[i].address.state);


        printf("\n\tNationality: %s\n", all_users[i].nationality);


        printf("\tMobile: %s\n", all_users[i].mobile);


        printf("\tEmail ID: %s\n", all_users[i].email);


        printf("\tDate of birth: %d-%d-%d\n", all_users[i].dob.day, all_users[i].dob.month,
all_users[i].dob.year);


        printf("\n\tAge: %d\n", all_users[i].age);


        printf("\n\tF: Female M: Male T:Transgender O:Other\n");
        printf("\tGender: %c\n", all_users[i].gender);


        i++;


        getch();
    }


    fclose(fptr);


}
```

```
/*
AIRLINE CODE            PASSWORD
123                      abc
456                     def
789                      ghi
*/
  FILE *fp;
 FILE *a, *f;
enum days {SUNDAY=1, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY};
enum places{CHENNAI, BANGALORE, PUNE, DELHI, KOCHI, MUMBAI, HYDERABAD, SINGAPORE,
LONDON, DUBAI};
typedef float price;
char place[][10]={"CHENNAI", "BANGALORE", "PUNE", "DELHI", "KOCHI", "MUMBAI", "HYDERABAD",
"SINGAPORE", "LONDON", "DUBAI"};
char day[][10]={"", "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"};

typedef struct
{
int code;
char pw[15];
}admin;

struct time
{
int hh;
int mm;
};

struct flight_det
```

```
{
int acode;                  //airline code
char fcode[10];
enum places source;
enum places destination;
struct time deptime;
struct time arrtime;
enum days day;
price adult_first;
price adult_business;
price adult_economy;
price child_first;
price child_business;
price child_economy;
};

struct flight
{
   struct flight_det details;
   struct date flight_date;
};

struct path
{
   struct flight route[30];
   int num; //no. of flights travelled in the route
   float cost;
};
```

```
void arr_cpy(int a[10], int b[10])
{
    for (int i=0; i<10; i++)
    {
        a[i]=b[i];
    }
}
void lower(char str[])

{   int i=0;
    while (str[i]!='\0')
    {
        str[i]=tolower(str[i]);
        i++;
    }
}



int isleap(int y)
{
    if (y%4==0)
        if (y%100==0)
            if (y%400==0)
                return 1;
            else
                return 0;
        else
            return 1;
    else
        return 0;


}
int datetoday(int d, int m, int y)
{
    //no. of days since the beginning of the year
    int month[12] = { 31, 28, 31, 30, 31, 30,
                      31, 31, 30, 31, 30, 31 };


    if (isleap(y))
        month[1]=29;
    int days=0;
    for (int i=0; i<m-1; i++)
    {
        days+=month[i];
    }


    days+=d;
    return days;
}


void addtodate(int x, int d, int m, int y, struct date *newdate)
{
    int days, total, rem;
    int month[12] = { 31, 28, 31, 30, 31, 30,
                      31, 31, 30, 31, 30, 31 };


    days= datetoday(d, m, y)+x;
```

```
total=isleap(y)?366:365;
if (isleap(y))
   month[1]=29;
rem=total-(days-x);
if (x>rem)
{
   newdate->year=y+1;
   days=x-rem;
   if (isleap(y))
      month[1]=29;
   else
      month[1]=28;
   newdate->month=1;
   int i=0;
   while(days>month[i])
   {
      days-=month[i];
      (newdate->month)++;
      i++;

   }
}

else
{
   newdate->year=y;
   newdate->month=1;
   int i=0;
   while(days>month[i])
```

```
        {
            days-=month[i];
            (newdate->month)++;
            i++;
         }
    }
    newdate->day=days;
}

int convert(int mm, int dd, int yy) /* convert date to numerical day of week */

{
long ndays; /* number of days from start of 1900 */
 long ncycles; /* number of 4-year cycles beyond 1900 */
 int nyears; /* number of years beyond last 4-year cycle */
 /* numerical conversions */
yy-=1900;
 ndays = (long) (30.42 * (mm - 1)) + dd; /* approximate day of year */

if (mm == 2) ++ndays; /* adjust for February */

if ((mm > 2) && (mm < 8)) --ndays; /* adjust for March - July */

if ((yy % 4 == 0) && (mm > 2)) ++ndays; /* adjust for leap year */

ncycles = yy / 4; /* 4-year cycles beyond 1900 */

ndays += ncycles * 1461; /* add days for 4-year cycles */
```

```
nyears = yy % 4; /* years beyond last 4-year cycle */

if (nyears > 0) /* add days for yrs beyond last 4-year cycle */

ndays += 365 * nyears + 1 ;

if (ndays > 59) --ndays; /* adjust for 1900 (NOT a leap year) */

return(ndays) ;

}

int timediff(int days, struct time arr, struct time dep)
{
   int result=(days*24*60) + (dep.hh-arr.hh)*60+(dep.mm-arr.mm);
   return result;
}

void printseats()
{
printf("\n                ______");
printf("\n               /  |  \\");
printf("\n              /___|___\\");
printf("\n             /         \\");
printf("\n            /            \\");
printf("\n           /              \\");
printf("\n          /                \\");
printf("\n                |   A B C    D E F   |");
printf("\n        <<  |  | | | | 1 | | | |  |  >>");
printf("\n        |  | | | | 2 | | | |  |                    FIRST CLASS");

printf("\n        |  | | | | 3 | | | |  |");

printf("\n               |                    |");

printf("\n           /|  | | | | 4 | | | |  |\\");

printf("\n         /  |  | | | | 5 | | | |  | \\                BUSINESS CLASS");

printf("\n       /    |  | | | | 6 | | | |  |  \\");

printf("\n         / <<  |                    |  >> \\");

printf("\n       /       |  | | | | 7 | | | |  |      \\");

printf("\n     /     |  | | | | 8 | | | |  |       \\");

printf("\n   /           |  | | | | 9 | | | |  |        \\");

printf("\n  /          /  |  | | | |10 | | | |  | \\        \\       ECONOMY CLASS");

printf("\n/        /    |  | | | |11 | | | |  |   \\       \\");

printf("\n|      /   <<  |  | | | |12 | | | |  | >>   \\      |");

printf("\n|____/          |  | | | |13 | | | |  |        \\____|");

printf("\n        |  | | | |14 | | | |  |");

printf("\n        |  | | | |15 | | | |  |\n");

}


void DFS(enum places source, enum places dest, enum places day, int visited[], struct path trip,int
*path_no, struct flight adj_list[][10], int nodes[], struct path paths[], struct date *dep_date, struct tm
*curr)

{

    visited[source]=1;

    int days, time;

    int v_cpy[10]; //copy of the visited array

    struct path trip_cpy;

    if (source==dest)

    {

        paths[*path_no]=trip;
        (*path_no)++;
    }

    else if (trip.num!=0)
    {
        for (int i=0; i<nodes[source]; i++)
        {
            if (visited[adj_list[source][i].details.destination]==0)
            {
                 if ((int)(adj_list[source][i].details.day-trip.route[trip.num-1].details.day)<0)
                    days=(adj_list[source][i].details.day-trip.route[trip.num-1].details.day+7);
                else
                    days=adj_list[source][i].details.day-trip.route[trip.num-1].details.day; //no. of days passes
between arrival of previous flight and departure of the current flight

                time=timediff(days,trip.route[trip.num-1].details.arrtime, adj_list[source][i].details.deptime);
                trip_cpy=trip;      //copy of trip
                trip_cpy.route[trip.num]=adj_list[source][i];
                trip_cpy.num++;
                addtodate(days, trip.route[trip.num-1].flight_date.day, trip.route[trip.num-
1].flight_date.month, trip.route[trip.num-1].flight_date.year, &trip_cpy.route[trip_cpy.num-
1].flight_date);//adds to days to date

                if (time>15&&time<=360&&(convert(trip_cpy.route[trip_cpy.num-1].flight_date.month,
trip_cpy.route[trip_cpy.num-1].flight_date.day, trip_cpy.route[trip_cpy.num-1].flight_date.year)-
convert(curr->tm_mday, curr->tm_mon+1, curr->tm_year+1900))<=30)
                //waiting time must be greater than 15 mins and lesser than 6 hours. Date must not be more
than 30 days from the current date
                {
                    arr_cpy(v_cpy, visited);
                DFS(adj_list[source][i].details.destination, dest, day, v_cpy, trip_cpy, path_no, adj_list, nodes,
paths, dep_date, curr);
            }
        }
    }
  }
  else
  {
    for (int i=0; i<nodes[source]; i++)
    {
      if (visited[adj_list[source][i].details.destination]==0&&adj_list[source][i].details.day==day)
        //For the firat flight, day of operation must be same as the day of departure

      {
        trip_cpy=trip;
        trip_cpy.route[trip.num]=adj_list[source][i];
        trip_cpy.route[trip.num].flight_date=*dep_date;
        trip_cpy.num++;
        arr_cpy(v_cpy, visited);
        DFS(adj_list[source][i].details.destination, dest, day, v_cpy, trip_cpy, path_no, adj_list, nodes,
paths, dep_date, curr);
      }
    }
  }
  return;
}

void path_cost(int p_array[][3], struct path *trip, float per)
{
    float total=0;

    for (int i=0; i<trip->num; i++)

    {

        total+= p_array[0][0]*trip->route[i].details.child_first+p_array[0][1]*trip-
>route[i].details.child_business+p_array[0][2]*trip->route[i].details.child_economy+

        p_array[1][0]*trip->route[i].details.adult_first+p_array[1][1]*trip-
>route[i].details.adult_business+p_array[1][2]*trip->route[i].details.adult_economy;

    }


    trip->cost=total*per/100;


}


void display(struct path paths[], int path_no, int p_arr[][3] )

{

    char dummy;

    if( path_no==0||paths[0].num==0)

        printf("\nNO FLIGHTS AVAILABLE\n\n");


    else

    {

        printf("\nFlIGHT ROUTES\n");

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

        {

            printf("%d.", i+1);


            for( int j=0; j<paths[i].num; j++)

            {

                printf("%s", cities[paths[i].route[j].details.source]);
                printf("->");
                if(j==paths[i].num-1)
                    printf("%s", cities[paths[i].route[j].details.destination]);
            }
            printf("\n");

            for( int j=0; j<paths[i].num; j++)
            {
                printf("\n\t%s->%s\n", cities[paths[i].route[j].details.source],
cities[paths[i].route[j].details.destination]);
                printf("\tAirline Code: %d\n\tFlight code: %s\n\tDeparture time: %d:%d\n\tArrival time:
%d:%d\n", paths[i].route[j].details.acode, paths[i].route[j].details.fcode,
paths[i].route[j].details.deptime.hh, paths[i].route[j].details.deptime.mm,
paths[i].route[j].details.arrtime.hh, paths[i].route[j].details.arrtime.mm);
                printf("\tDate: %d/%d/%d\n", paths[i].route[j].flight_date.day,
paths[i].route[j].flight_date.month, paths[i].route[j].flight_date.year);
                scanf("%c", &dummy);

            }
            printf("\n");
            printf("Total cost of path: %.2f", paths[i].cost);

            scanf("%c", &dummy);

        }

    }

}
```

```
int is_direct(struct path *trip)
{
   return trip->num==1;
}

int single(struct path *trip)
{
   return trip->num==2;
}

int multi(struct path *trip)
{
   return trip->num>=3;
}
int no_filter(struct path *trip)
{
   return 1;
}
void filter(struct path paths[], int (*f)(struct path*),  int* path_no)
{
   int count=0;
   for (int i=0; i<*path_no; i++)
   {
      if ((*f)(&paths[i]))
      {
         paths[count]=paths[i];
         count++;
      }
```

```
    }
    *path_no=count;

}
int lcost(struct path *trip1, struct path *trip2)
{
    return trip1->cost<trip2->cost;
}

int hcost(struct path *trip1, struct path *trip2)
{
    return trip1->cost>trip2->cost;
}

int stops(struct path *trip1, struct path *trip2)
{
    return trip1->num<trip2->num;
}

void sort(struct path paths[], int path_no, int (*f)(struct path*, struct path*))
{   int min_index;
    for (int i=0; i<path_no-1; i++)
    {
        min_index=i;
        for(int j=i+1; j<path_no; j++)
        {
            if ((*f)(&paths[j], &paths[min_index]))
                min_index=j;
        }


        struct path temp=paths[min_index];

        paths[min_index]=paths[i];

        paths[i]=temp;


    }

}


void book()

{

    //f=fopen("a.bin", "rb+");

    int n; //no. of passengers

    FILE *f;

    float per;

    time_t timer;

    struct tm curr_time;

    int diff;

    struct date dep_date;

    while(1)

    {

       time(&timer);

        curr_time=*localtime(&timer);  //Current time

        printf("\nEnter date of departure as day/month/year: ");

        scanf(" %d/%d/%d", &dep_date.day, &dep_date.month, &dep_date.year);

        diff=convert(dep_date.month, dep_date.day,  dep_date.year)-convert(curr_time.tm_mon+1,
curr_time.tm_mday, curr_time.tm_year+1900);

        if (diff<0)

            printf("\nEntered date is past current date.\nPlease re-enter.\n");
        else if (diff>30)
            printf("\nBooking is available for uptil 30 days before the flight journey.\nPlease re-enter.\n");
        else
            break;
    }

    enum days day=convert( dep_date.month, dep_date.day, dep_date.year)%7+1;
    if (diff>20)
        per=90;
    else if(diff>10)
        per=95;
    else
        per=100;

    int p_arr[][3]={{0, 0, 0}, {0, 0, 0}}; //Stores no. of passengers in each category
    enum places source;
    enum places dest;

printf("\n\nSOURCE\n\nOptions:\n0.CHENNAI\n1.BANGALORE\n2.PUNE\n3.DELHI\n4.KOCHI\n5.MUMB
AI\n6.HYDERABAD\n7.SINGAPORE\n8.LONDON\n9.DUBAI\n\n");
    printf("Enter option: ");
    scanf(" %u",&source);

printf("\n\nDESTINATION\nOptions:\n0.CHENNAI\n1.BANGALORE\n2.PUNE\n3.DELHI\n4.KOCHI\n5.MU
MBAI\n6.HYDERABAD\n7.SINGAPORE\n8.LONDON\n9.DUBAI\n\n");
    printf("Enter option: ");
    scanf(" %u",&dest);

    system("clear");
```

```
int choice1, choice2;


printf("Enter no. of passengers: ");

scanf("%d", &n);


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

{

   printf("Passenger %d\n", i+1);

   printf("\n\nChoose Age Category\n1. Child(Lesser than 12yrs)\n2. Adult\n\nEnter: ");

   scanf("%d", &choice1);

   if (choice1!=1&&choice1!=2)

   {

      printf("Invalid Choice.\nPlease Re-enter.\n");

      i--;

      continue;

   }

   while(1)

   {

      printf("\n\nChoose Class\n1. First Class\n2. Business Class\n3. Economy Class\n\nEnter: ");

      scanf("%d", &choice2);

      if (choice2!=1&&choice2!=2&&choice2!=3)

         printf("Invalid Choice.\nPlease Re-enter.\n");

      else

         break;


   }


   system("clear");
```

```
  p_arr[choice1-1][choice2-1]++;
}
struct path paths[1024]; //Stores all possible paths from source to destination
struct path trip;      //Stores a particular path
trip.num=0;            //No. of flights taken in trip
int path_no=0;         //Total no. of paths from source->destination
int visited[10];       //To keep track of visited places to avoid cycles in the graph
for (int i=0; i<10; i++)
{
  visited[i]=0;      //Marking all places as unvisited
}

struct flight flights[30];
int num=0; //no. of available flights
for (int i=0; i<3; i++)
{
  f=fopen(fnames[i], "rb");
  while(fread(&flights[num].details, sizeof(struct flight_det), 1, f))
  {
    num++;
  }
  fclose(f);
}

int nodes[10]; //no. of flights from each node in the graph
for (int i=0; i<10; i++)
{
  nodes[i]=0;
}
```

```
    struct flight adj_list[10][10]; //adjacency list
    for (int i=0; i<num; i++)
    {
        adj_list[flights[i].details.source][nodes[flights[i].details.source]]=flights[i];
        nodes[flights[i].details.source]+=1;
    }

    DFS(source, dest, day, visited, trip, &path_no, adj_list, nodes, paths, &dep_date, &curr_time);

    for (int i=0; i<path_no; i++)
    {
        path_cost(p_arr, &paths[i], per);
    }
    display(paths, path_no, p_arr);

    if(path_no!=0&&paths[0].num!=0)
   {
        system("clear");

        printf("Enter filter criteria\n1.Direct flights\n2.Single stop routes\n3.Atleast 2 stops\n4.No applied
filter\nEnter: ");
        scanf("%d", &choice1);
        printf("Sort based on\n1.Cost(low to high)\n2.Cost(high to low)\n3.No. of stops(low to
high)\nEnter: ");

        scanf("%d", &choice2);
        int (*fil)(struct path*);
        int (*s)(struct path*, struct path*);
        switch(choice1)

        {

           case 1: fil=&is_direct;break;

           case 2: fil=&single;break;

           case 3: fil=&multi;break;

           case 4: fil=&no_filter;break;


        }


        switch(choice2)

        {

           case 1: s=&lcost;break;

           case 2: s=&hcost;break;

           case 3: s=&stops;break;

        }


        filter(paths, fil,  &path_no);

        sort(paths, path_no, s);


        display(paths, path_no, p_arr);


        system("clear");


        printf("\n\nSEATING ARRANGEMENT\n\n");

        printseats();


    }

}
```

```
void user_menu()
{
    system("clear");

    char ch;

    printf("\n\n\n\tUSER OPTIONS");
    printf("\n\n\tTICKET OPTIONS");
    printf("\n\n\t\t1. Book Flight Ticket");
    printf("\n\n\t\t2. Current Flight Ticket Bookings");
    printf("\n\n\t\t3. Cancel Flight Ticket");
    printf("\n\n\tACCOUNT OPTIONS");
    printf("\n\n\t\t4. View Account Profile");
    printf("\n\n\t\t5. Update Account Profile");
    printf("\n\n\t\t6. Delete Account");
    printf("\n\n\tEXIT OPTION");
    printf("\n\n\t\t7. BACK TO MAIN MENU");

    printf("\n\n\n\n\tEnter choice: ");
    scanf(" %c", &ch);

    switch(ch)

    {
        case '1':
                system("clear");
                book();
                getch();
        user_menu();
        break;

case '2':    system("clear");
        ticket_enquiry();
        user_menu();
        break;

case '3':    system("clear");
        getch();
        user_menu();
        break;

case '4':    system("clear");
        user_det_view_particular();
        user_menu();
        break;

case '5':    system("clear");
        user_det_update();
        user_menu();
        break;

case '6':    system("clear");
        user_det_delete();
        break;

case '7':    system("clear");
        break;
        default:    user_menu();
    }
}
 void add_flights(int n, int acode)
{
struct flight_det flight;
int day;
for (int i=0;i<n;i++)
{
flight.acode=acode;
printf("\nEnter flight code: ");
scanf(" %5s",flight.fcode);
do
{
printf("Source Options:\n\t0. CHENNAI\n\t1. BANGALORE\n\t2. PUNE\n\t3. DELHI\n\t4. KOCHI\n\t5.
MUMBAI\n\t6. HYDERABAD\n\t7. SINGAPORE\n\t8. LONDON\n\t9. DUBAI\n");
printf("Enter option: ");
scanf(" %u",&flight.source);
}while(flight.source<0 || flight.source>9);
do
{
printf("Destination Options:\n\t0. CHENNAI\n\t1. BANGALORE\n\t2. PUNE\n\t3. DELHI\n\t4.
KOCHI\n\t5. MUMBAI\n\t6. HYDERABAD\n\t7. SINGAPORE\n\t8. LONDON\n\t9. DUBAI\n");
printf("Enter option: ");
scanf(" %u",&flight.destination);
}while(flight.destination<0 || flight.destination>9);
do
{
```

```
printf("Enter departure time(hh/mm) : ");

scanf("%d/%d",&flight.deptime.hh,&flight.deptime.mm);

}while(flight.deptime.hh<0 || flight.deptime.hh>24 || flight.deptime.mm<0 || flight.deptime.mm>60);

do

{

printf("Enter arrival time(hh/mm)   : ");

scanf("%d/%d",&flight.arrtime.hh,&flight.arrtime.mm);

}while(flight.arrtime.hh<0 || flight.arrtime.hh>24 || flight.arrtime.mm<0 || flight.arrtime.mm>60);

do

{

printf("Day of operation of flight option:\n\t1. Sundays\n\t2. Mondays\n\t3. Tuesdays\n\t4.
Wednesdays\n\t5. Thursdays\n\t6. Fridays\n\t7. Saturdays\nEnter choice: ");

scanf("%d",&day);

switch (day)

{

case 1: flight.day=SUNDAY;

break;

case 2: flight.day=MONDAY;

break;

case 3: flight.day=TUESDAY;

break;

case 4: flight.day=WEDNESDAY;

break;

case 5: flight.day=THURSDAY;

break;

case 6: flight.day=FRIDAY;

break;

case 7: flight.day=SATURDAY;

break;
}
}while(day<1 || day>7);
printf("Adult ticket prices\n");
printf("\tEnter for first class    : ");
scanf("%f",&flight.adult_first);
printf("\tEnter for business class : ");
scanf("%f",&flight.adult_business);
printf("\tEnter for economy class  : ");
scanf("%f",&flight.adult_economy);
printf("Child ticket prices\n");
printf("\tEnter for first class    : ");
scanf("%f",&flight.child_first);
printf("\tEnter for business class : ");
scanf("%f",&flight.child_business);
printf("\tEnter for economy class  : ");
scanf("%f",&flight.child_economy);

fwrite(&flight, sizeof(struct flight_det), 1, f);
}
}

int read_det(int acode, struct flight_det allflight[])
{
int count;
fseek(f, 0, SEEK_END);
count=ftell(f)/sizeof(struct flight_det);
fseek(f, 0, SEEK_SET);
fread(allflight, sizeof(struct flight_det), count, f);
return count;
```

```
}

void disp(int i,struct flight_det allflight[])

{

printf("%-5d %-5s %-10s %-10s   %02d:%02d     %02d:%02d   %-9s
%10.2f%10.2f%10.2f%10.2f%10.2f%10.2f\n", allflight[i].acode, allflight[i].fcode, place[allflight[i].source],
place[allflight[i].destination], allflight[i].deptime.hh, allflight[i].deptime.mm, allflight[i].arrtime.hh,
allflight[i].arrtime.mm, day[allflight[i].day], allflight[i].adult_first, allflight[i].adult_business,
allflight[i].adult_economy, allflight[i].child_first, allflight[i].child_business, allflight[i].child_economy);

}


int search(int n, struct flight_det allflight[])

{

char search[20];

printf("Enter flight code: ");

scanf(" %s",search);

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

{

   if(strcmp(allflight[i].fcode,search)==0)

       return i;

}

return -1;

}


void modify(int index,int n,struct flight_det allflight[])

{

int day;

printf("Source Options:\n\t0. CHENNAI\n\t1. BANGALORE\n\t2. PUNE\n\t3. DELHI\n\t4. KOCHI\n\t5.
MUMBAI\n\t6. HYDERABAD\n\t7. SINGAPORE\n\t8. LONDON\n\t9. DUBAI\n");

printf("Enter new option: ");

scanf(" %u",&allflight[index].source);
```

```
printf("Destination Options:\n\t0. CHENNAI\n\t1. BANGALORE\n\t2. PUNE\n\t3. DELHI\n\t4.
KOCHI\n\t5. MUMBAI\n\t6. HYDERABAD\n\t7. SINGAPORE\n\t8. LONDON\n\t9. DUBAI\n");

printf("Enter new option: ");

scanf(" %u",&allflight[index].destination);

printf("Enter new departure time(hh/mm) : ");

scanf("%d/%d",&allflight[index].deptime.hh,&allflight[index].deptime.mm);

printf("Enter new arrival time(hh/mm)   : ");

scanf("%d/%d",&allflight[index].arrtime.hh,&allflight[index].arrtime.mm);

printf("Day of operation of flight option:\n\t1. Sundays\n\t2. Mondays\n\t3. Tuesdays\n\t4.
Wednesdays\n\t5. Thursdays\n\t6. Fridays\n\t7. Saturdays\nEnter new choice: ");

scanf("%d",&day);

switch (day)

   {

   case 1: allflight[index].day=SUNDAY;

   break;

   case 2: allflight[index].day=MONDAY;

   break;

   case 3: allflight[index].day=TUESDAY;

   break;

   case 4: allflight[index].day=WEDNESDAY;

   break;

   case 5: allflight[index].day=THURSDAY;

   break;

   case 6: allflight[index].day=FRIDAY;

   break;

   case 7: allflight[index].day=SATURDAY;

   break;

   }

printf("Adult new ticket prices\n");
printf("\tEnter for first class    : ");
scanf("%f",&allflight[index].adult_first);
printf("\tEnter for business class : ");
scanf("%f",&allflight[index].adult_business);
printf("\tEnter for economy class  : ");
scanf("%f",&allflight[index].adult_economy);
printf("Child new ticket prices\n");
printf("\tEnter for first class    : ");
scanf("%f",&allflight[index].child_first);
printf("\tEnter for business class : ");
scanf("%f",&allflight[index].child_business);
printf("\tEnter for economy class  : ");
scanf("%f",&allflight[index].child_economy);

fseek(f, 0, SEEK_SET);
fwrite(allflight, sizeof(struct flight_det), n, f);
}

void delete(int index,int n,struct flight_det allflight[])
{
for (int i=index;i<n;i++)
   allflight[i]=allflight[i+1];
fwrite(allflight,sizeof(struct flight_det),--n,f);
printf("Flight Record Deleted!!\n");
}

void admin_menu(int acode)
{
   system("clear");
```

```
char ch, file[10];

int n,index;

struct flight_det flight, allflight[20];

if (acode==123)

strcpy(file,"123.bin");

else if (acode==456)

strcpy(file,"456.bin");

else if (acode==789)

strcpy(file,"789.bin");


printf("\n\n\n\tADMINISTRATOR OPTIONS");

printf("\n\n\tFLIGHT OPTIONS");

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

printf("\n\n\t\t2. Update Existing Flight");

printf("\n\n\t\t3. Delete A Particular Flight");

printf("\n\n\t\t4. Display A Particular Flight");

printf("\n\n\t\t5. Display All Flights");

printf("\n\n\tCUSTOMER OPTIONS");

printf("\n\n\t\t6. View A Particular User Account Profile");

printf("\n\n\t\t7. View All User Account Profiles");

printf("\n\n\tEXIT OPTION");

printf("\n\n\t\t8. BACK TO MAIN MENU");


printf("\n\n\n\n\tEnter choice: ");

scanf(" %c", &ch);


switch(ch)
```

```
{
   case '1':    system("clear");
   f=fopen(file,"ab+");
   printf("Enter number of flights to be added: ");
   scanf("%d",&n);
   add_flights(n,acode);
   fclose(f);

         admin_menu(acode);
         break;

   case '2':    system("clear");
         f=fopen(file,"rb+");
   n=read_det(acode, allflight);
   index=search(n, allflight);
   if (index==-1)
      printf("\nFlight details not found!!\n");
   else
      modify(index,n,allflight);
   fclose(f);
         admin_menu(acode);
         break;

   case '3':    system("clear");
   f=fopen(file,"rb");
   n=read_det(acode, allflight);
   index=search(n, allflight);
   if (index==-1)
      printf("\nFlight details not found!!\n");
```

```
        else
          {
          f=fopen(file,"wb");
          delete(index,n,allflight);
          }
        fclose(f);
              admin_menu(acode);
              break;


        case '4':    system("clear");
              f=fopen(file,"rb");
        n=read_det(acode, allflight);
        index=search(n, allflight);
        if (index==-1)
          printf("\nFlight code does not exist!!\n");
        else
        {
printf("ACODE FCODE SOURCE    DESTINATION  DEPARTURE ARRIVAL    DAY              ADULT FARES
CHILD FARES\n");
printf("                                                   FIRST    BUSINESS  ECONOMY     FIRST    BUSINESS
ECONOMY\n");
          disp(index, allflight);
        }
        getch();
        fclose(f);
        admin_menu(acode);
          break;


        case '5':    system("clear");
```

```
            f=fopen(file,"rb");
        n=read_det(acode, allflight);
printf("ACODE FCODE SOURCE    DESTINATION  DEPARTURE ARRIVAL    DAY              ADULT FARES
CHILD FARES\n");
printf("                                                FIRST    BUSINESS  ECONOMY     FIRST    BUSINESS
ECONOMY\n");
        for (int i=0;i<n;i++)
        {
          disp(i,allflight);
        }
        getch();
        fclose(f);
        admin_menu(acode);
            break;


        case '6':    system("clear");
          user_det_view_particular();
          admin_menu(acode);
          break;



        case '7':    system("clear");
          user_det_view_all();
          admin_menu(acode);
          break;



        case '8':    system("clear");
              break;
        default:    admin_menu(acode);
    }
}

/*  To get the password from the keyboard.
Uses pass by reference to get the entered password
via parameter to function  */

void getPassword(char *pass)
{
    char ch;
      int len=0;

    while((ch=getch())!='\n')
      {
        printf("*");

        pass[len]=ch;
        len++;
      }

    pass[len]='\0';
}

void admin_add()
{
a=fopen("admin_det.bin","wb");
admin alldet[3];
```

```
alldet[0].code=123;

strcpy(alldet[0].pw,"abc");

alldet[1].code=456;

strcpy(alldet[1].pw,"def");

alldet[2].code=789;

strcpy(alldet[2].pw,"ghi");

fwrite(alldet, sizeof(admin), 3, a);

fclose(a);

}


int check(admin det)

{

int count;

a=fopen("admin_det.bin","rb");

fseek(a, 0, SEEK_END);

count=ftell(a)/sizeof(admin);

admin alldet[count];

fseek(a, 0, SEEK_SET);

fread(alldet, sizeof(admin), count, a);

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

    if (det.code==alldet[i].code && strcmp(det.pw,alldet[i].pw)==0)

        return det.code;

return 0;

}


void admin_login()

{

    system("clear");

    admin det;
    int user;
    char pass[30];

    printf("\n\n\n\t\tADMIN LOGIN");
    printf("\n\n\n\t\tAirline code: ");
    scanf(" %d", &det.code);
    printf("\n\t\tPassword: ");
    getPassword(det.pw);

    if (check(det))
    admin_menu(det.code);

    else
    {
        printf("\n\n\n\t\tLOGIN FAILED....!!!!");
        getch();
    }
}

void user_det_input()
{
    FILE *fptr= fopen("user.bin", "ab");
    struct user_details user;

    printf("\n\n\n\tSIGNUP\n\n");
    printf("\tNEW USER ACCOUNT ENTRY\n\n");

    printf("\tUSERNAME: ");
```

```
scanf(" %[^\n]", user.username);

printf("\tPASSWORD: ");
scanf(" %[^\n]", user.password);

printf("\n\tName: ");
scanf(" %[^\n]", user.name);

printf("\n\tAddress:\n");
printf("\tStreet: ");
scanf(" %[^\n]", user.address.street);

printf("\tCity: ");
scanf(" %[^\n]", user.address.city);

printf("\tPincode: ");
scanf(" %[^\n]", user.address.pincode);

printf("\tState: ");
scanf(" %[^\n]", user.address.state);

printf("\n\tNationality: ");
scanf(" %[^\n]", user.nationality);

printf("\tMobile: ");
scanf(" %[^\n]", user.mobile);

printf("\tEmail ID: ");
scanf(" %[^\n]", user.email);
```

```
    printf("\n\tEnter DATE in dd mm yyyy FORMAT\n");
    printf("\tDate of birth: ");
    scanf("%d %d %d", &user.dob.day, &user.dob.month, &user.dob.year);

    printf("\n\tAge: ");
    scanf("%d", &user.age);

    printf("\n\tF: Female M: Male T:Transgender O:Other\n");
    printf("\tGender: ");
    scanf(" %c", &user.gender);

    fwrite(&user, sizeof(struct user_details), 1, fptr);

    fseek(fptr, 0, SEEK_END);

    fclose(fptr);

     printf("\n\n\n\tNEW USER ACCOUNT ENTRY SUCCESSFULL......!!!!!\n");

    getch();

}

void user_login()
{
    system("clear");

    char pass[30];
```

```
struct user_details all_users[10];
int count=0, index=-1;

printf("\n\n\n\tUSER LOGIN");

count=read_count_users(all_users);

printf("\n");

index=search_users(count, all_users);

if(index==-1)
{
   printf("\n\n\n\tUSERNAME INVALID....!!!!");
   getch();
}

else
{
   printf("\n\tPassword: ");
   getPassword(pass);

   if(strcmp(all_users[index].password, pass)==0)
      user_menu();

   else
   {
      printf("\n\n\n\tLOGIN FAILED....!!!!");
      getch();
```

```
        }

    }

}

void user_terminal()
{
    char ch;

    system("clear");

    do
    {   system("clear");

        printf("\n\n\n\tSNL AIRLINE BOOKING");
        printf("\n\n\t1. LOGIN");
        printf("\n\n\t2. SIGNUP");
        printf("\n\n\t3. BACK TO MAIN MENU ");

        printf("\n\n\n\n\tEnter choice: ");
        scanf(" %c", &ch);

        switch(ch)
        {
            case '1':   system("clear");
                    user_login();
                    break;
            case '2':    system("clear");
                    user_det_input();
                    break;

            case '3':    system("clear");
                    break;
        }
    }while(ch!='3');

}

int main()
{
    char ch;
    intro();
    admin_add();

    do
    {    system("clear");

        printf("\n\n\n\tMAIN MENU");
        printf("\n\n\t1. USER");
        printf("\n\n\t2. ADMINISTRATOR");
        printf("\n\n\t3. EXIT");

        printf("\n\n\n\n\tEnter choice: ");
        scanf(" %c", &ch);

        switch(ch)
        {
            case '1':    system("clear");
                    user_terminal();
                    break;

            case '2':    system("clear");
                    admin_login();
                    break;

            case '3':    return 0;
        }
    }while(ch!='3');
}
```