#include<iostream>

#include<cstring>

using namespace std;

typedef struct patient

{

char doctornam[50][10];

char time[50][9],month[50][2],year[50][4];

int date[50];

struct patient \*next;

}\*node;

class reg

{

int que;

char name[20],username[15],password[15],gmail[30],phone[10],ans[20];

public:

int task\_num=0;

void add\_details();

void incr\_task();

friend void forgot();

friend int login();

friend int search(char user1[15]);

friend int search1(char u1[15],char p1[15]);

friend void add\_task(node x);

friend void check\_appointment(int m);

};

static char doctor\_name[8][10]={"Shilpa","Muskan","Ajay","Arjun","Neha","Shraddha","Anjali","\0"};

const int size = 50;

static reg s[size];

static char doctor\_des[8][20]={ "HeartSpecialist" , "Gynaelogist", "Orthopedic ",

"Neurologist" ,"Dentist" ,"Pediatric","GeneralPhysician"};

static char doctor\_age[8][3]={ "32" ,"35", "42","47" ,"49" ,"52","55"};

static char doctor\_num[8][11]={ "3278543212" ,"8999202790", "4246588553","8638764047" ,"1324555249" ,"6546444152","9809786155"};

static char doctor\_user[8][6]={ "user1" ,"user2", "user3","user4" ,"user5" ,"user6","user7"};

static char doctor\_pass[8][10]={ "passuser1" ,"passuser2", "passuser3","passuser4" ,"passuser5" ,"passuser6","passuser7"};

class doctor

{

char name1[15],designation[25],phone[10],age[3],dusername[6],dpassword[10];

public:

friend void doctor\_details();

int patient\_check[50],total\_appointment=0,task\_count[50];

friend void appointment\_count(int x,int y,int z);

friend int dsearch1(char u1[15],char p1[15]);

friend int dlogin();

friend void check\_appointment(int m);

};

const int size1 = 8;

static doctor d[size1];

static node first[50];

void appointment\_count(int x,int y,int z)

{

int a=d[x].total\_appointment;

d[x].patient\_check[a]=y;

d[x].task\_count[a]=z;

++d[x].total\_appointment;

}

void check\_appointment(int m)

{

int i,patient\_id,patient\_task,j,t,t1;

node a;

cout<<"\n No \t Patient name \t Patient date \t Patient time ";

for(i=0;i<d[m].total\_appointment;i++)

{

patient\_id=d[m].patient\_check[i];

patient\_task=d[m].task\_count[i];

cout<<"\n"<<i+1;

t=d[m].task\_count[i];

t1=d[m].task\_count[i];

a=first[patient\_id];

while(t>1)

{

a=a->next;

--t;

}

cout<<"\t"<<s[patient\_id].name<<"\t"<<a->date[t1]<<"/"<<a->month[t1]<<"/"<<a->year[t1]<<"\t"<<a->time[t1];

}

}

void doctor\_details()

{

strcpy(d[0].name1,doctor\_name[0]);

strcpy(d[1].name1,doctor\_name[1]);

strcpy(d[2].name1,doctor\_name[2]);

strcpy(d[3].name1,doctor\_name[3]);

strcpy(d[4].name1,doctor\_name[4]);

strcpy(d[5].name1,doctor\_name[5]);

strcpy(d[6].name1,doctor\_name[6]);

strcpy(d[0].age,doctor\_age[0]);

strcpy(d[1].age,doctor\_age[1]);

strcpy(d[2].age,doctor\_age[2]);

strcpy(d[3].age,doctor\_age[3]);

strcpy(d[4].age,doctor\_age[4]);

strcpy(d[5].age,doctor\_age[5]);

strcpy(d[6].age,doctor\_age[6]);

strcpy(d[0].designation,doctor\_des[0]);

strcpy(d[1].designation,doctor\_des[1]);

strcpy(d[2].designation,doctor\_des[2]);

strcpy(d[3].designation,doctor\_des[3]);

strcpy(d[4].designation,doctor\_des[4]);

strcpy(d[5].designation,doctor\_des[5]);

strcpy(d[6].designation,doctor\_des[6]);

strcpy(d[0].phone,doctor\_num[0]);

strcpy(d[1].phone,doctor\_num[1]);

strcpy(d[2].phone,doctor\_num[2]);

strcpy(d[3].phone,doctor\_num[3]);

strcpy(d[4].phone,doctor\_num[4]);

strcpy(d[5].phone,doctor\_num[5]);

strcpy(d[6].phone,doctor\_num[6]);

strcpy(d[0].dusername,doctor\_user[0]);

strcpy(d[1].dusername,doctor\_user[1]);

strcpy(d[2].dusername,doctor\_user[2]);

strcpy(d[3].dusername,doctor\_user[3]);

strcpy(d[4].dusername,doctor\_user[4]);

strcpy(d[5].dusername,doctor\_user[5]);

strcpy(d[6].dusername,doctor\_user[6]);

strcpy(d[0].dpassword,doctor\_pass[0]);

strcpy(d[1].dpassword,doctor\_pass[1]);

strcpy(d[2].dpassword,doctor\_pass[2]);

strcpy(d[3].dpassword,doctor\_pass[3]);

strcpy(d[4].dpassword,doctor\_pass[4]);

strcpy(d[5].dpassword,doctor\_pass[5]);

strcpy(d[6].dpassword,doctor\_pass[6]);

}

static char slottime[12][8]={"10:00am","10:15am","10:30am","10:45am","11:00am","11:15am","11:30am","11:45am","12:00pm","12:15pm","12:30pm","12:45pm"};

static char slottime1[12][8]={"5:00pm","5:15pm","5:30pm","5:45pm","6:00pm","6:15pm","6:30pm","6:45pm","7:00pm","7:15pm","7:30pm","7:45pm"};

static int doctor\_schedule1[13];

static int doctor\_schedule2[13];

static int doctor\_schedule3[13];

static int doctor\_schedule4[13];

static int doctor\_schedule5[13];

static int doctor\_schedule6[13];

static int doctor\_schedule7[13];

static int doctor\_schedule11[13];

static int doctor\_schedule21[13];

static int doctor\_schedule31[13];

static int doctor\_schedule41[13];

static int doctor\_schedule51[13];

static int doctor\_schedule61[13];

static int doctor\_schedule71[13];

static int k1=1,k2=1,k3=1,k4=1,k5=1,k6=1,k7=1;

static int k11=1,k21=1,k31=1,k41=1,k51=1,k61=1,k71=1;

static int n=1;

void reg:: incr\_task()

{

int \*p;

p=&task\_num;

++\*p;

}

void fill\_details(node x,int n1)

{

int k,num,doption,dat;

k=s[n1].task\_num;

cout<<"\n Select the doctor \n 1 Dr Shilpa : Heart Specialist \n 2 Dr Muskan : Gynaelogist \n 3 Dr Ajay : Orthopedic ";

cout<<"\n 4 Dr Arjun : Neurologist \n 5 Dr Neha : Dentist \n 6 Dr Shraddha : Pediatric \n 7 Dr Anjali :General Physician";

cout<<"\n Enter the choice number of doctor with whom you want appointment ";

cin>>doption;

doption=doption-1;

appointment\_count(doption,n1,k);

strcpy(x->doctornam[k],doctor\_name[doption]);

cout<<"\n Enter the date";

cin>>x->date[k];

dat=x->date[k];

cout<<"\n Enter the month";

cin>>x->month[k];

cout<<"\n Enter the year";

cin>>x->year[k];

cout<<"\n Which slot do you want?\n Morning :10:00am to 1:00pm\n Evening :5:00pm to 8:00pm \n Press 1 :Morning \n Press 2 :Evening";

cin>>num;

if(num==1)

{

switch(doption)

{

case 1 : doctor\_schedule1[k1]=1;

switch(k1)

{

case 1 :

strcpy(x->time[k],slottime[0]);

break;

case 2 :

strcpy(x->time[k],slottime[1]);

break;

case 3 :

strcpy(x->time[k],slottime[2]);

break;

case 4 :

strcpy(x->time[k],slottime[3]);

break;

case 5 :

strcpy(x->time[k],slottime[4]);

break;

case 6 :

strcpy(x->time[k],slottime[5]);

break;

case 7 :

strcpy(x->time[k],slottime[6]);

break;

case 8 :

strcpy(x->time[k],slottime[7]);

break;

case 9 :

strcpy(x->time[k],slottime[8]);

break;

case 10 :

strcpy(x->time[k],slottime[9]);

break;

case 11 :

strcpy(x->time[k],slottime[10]);

break;

case 12 :

strcpy(x->time[k],slottime[11]);

break;

}

++k1;

break;

case 2 : doctor\_schedule2[k2]=1;

switch(k2)

{

case 1 :

strcpy(x->time[k],slottime[0]);

break;

case 2 :

strcpy(x->time[k],slottime[1]);

break;

case 3 :

strcpy(x->time[k],slottime[2]);

break;

case 4 :

strcpy(x->time[k],slottime[3]);

break;

case 5 :

strcpy(x->time[k],slottime[4]);

break;

case 6 :

strcpy(x->time[k],slottime[5]);

break;

case 7 :

strcpy(x->time[k],slottime[6]);

break;

case 8 :

strcpy(x->time[k],slottime[7]);

break;

case 9 :

strcpy(x->time[k],slottime[8]);

break;

case 10 :

strcpy(x->time[k],slottime[9]);

break;

case 11 :

strcpy(x->time[k],slottime[10]);

break;

case 12 :

strcpy(x->time[k],slottime[11]);

break;

}

++k2;

break;

case 3 : doctor\_schedule3[k3]=1;

switch(k3)

{

case 1 :

strcpy(x->time[k],slottime[0]);

break;

case 2 :

strcpy(x->time[k],slottime[1]);

break;

case 3 :

strcpy(x->time[k],slottime[2]);

break;

case 4 :

strcpy(x->time[k],slottime[3]);

break;

case 5 :

strcpy(x->time[k],slottime[4]);

break;

case 6 :

strcpy(x->time[k],slottime[5]);

break;

case 7 :

strcpy(x->time[k],slottime[6]);

break;

case 8 :

strcpy(x->time[k],slottime[7]);

break;

case 9 :

strcpy(x->time[k],slottime[8]);

break;

case 10 :

strcpy(x->time[k],slottime[9]);

break;

case 11 :

strcpy(x->time[k],slottime[10]);

break;

case 12 :

strcpy(x->time[k],slottime[11]);

break;

}

++k3;

break;

case 4 : doctor\_schedule4[k4]=1;

switch(k4)

{

case 1 :

strcpy(x->time[k],slottime[0]);

break;

case 2 :

strcpy(x->time[k],slottime[1]);

break;

case 3 :

strcpy(x->time[k],slottime[2]);

break;

case 4 :

strcpy(x->time[k],slottime[3]);

break;

case 5 :

strcpy(x->time[k],slottime[4]);

break;

case 6 :

strcpy(x->time[k],slottime[5]);

break;

case 7 :

strcpy(x->time[k],slottime[6]);

break;

case 8 :

strcpy(x->time[k],slottime[7]);

break;

case 9 :

strcpy(x->time[k],slottime[8]);

break;

case 10 :

strcpy(x->time[k],slottime[9]);

break;

case 11 :

strcpy(x->time[k],slottime[10]);

break;

case 12 :

strcpy(x->time[k],slottime[11]);

break;

}

++k4;

break;

case 5 : doctor\_schedule5[k5]=1;

switch(k5)

{

case 1 :

strcpy(x->time[k],slottime[0]);

break;

case 2 :

strcpy(x->time[k],slottime[1]);

break;

case 3 :

strcpy(x->time[k],slottime[2]);

break;

case 4 :

strcpy(x->time[k],slottime[3]);

break;

case 5 :

strcpy(x->time[k],slottime[4]);

break;

case 6 :

strcpy(x->time[k],slottime[5]);

break;

case 7 :

strcpy(x->time[k],slottime[6]);

break;

case 8 :

strcpy(x->time[k],slottime[7]);

break;

case 9 :

strcpy(x->time[k],slottime[8]);

break;

case 10 :

strcpy(x->time[k],slottime[9]);

break;

case 11 :

strcpy(x->time[k],slottime[10]);

break;

case 12 :

strcpy(x->time[k],slottime[11]);

break;

}

++k5;

break;

case 6 : doctor\_schedule6[k6]=1;

switch(k6)

{

case 1 :

strcpy(x->time[k],slottime[0]);

break;

case 2 :

strcpy(x->time[k],slottime[1]);

break;

case 3 :

strcpy(x->time[k],slottime[2]);

break;

case 4 :

strcpy(x->time[k],slottime[3]);

break;

case 5 :

strcpy(x->time[k],slottime[4]);

break;

case 6 :

strcpy(x->time[k],slottime[5]);

break;

case 7 :

strcpy(x->time[k],slottime[6]);

break;

case 8 :

strcpy(x->time[k],slottime[7]);

break;

case 9 :

strcpy(x->time[k],slottime[8]);

break;

case 10 :

strcpy(x->time[k],slottime[9]);

break;

case 11 :

strcpy(x->time[k],slottime[10]);

break;

case 12 :

strcpy(x->time[k],slottime[11]);

break;

}

++k6;

break;

case 7 : doctor\_schedule7[k7]=1;

switch(k7)

{

case 1 :

strcpy(x->time[k],slottime[0]);

break;

case 2 :

strcpy(x->time[k],slottime[1]);

break;

case 3 :

strcpy(x->time[k],slottime[2]);

break;

case 4 :

strcpy(x->time[k],slottime[3]);

break;

case 5 :

strcpy(x->time[k],slottime[4]);

break;

case 6 :

strcpy(x->time[k],slottime[5]);

break;

case 7 :

strcpy(x->time[k],slottime[6]);

break;

case 8 :

strcpy(x->time[k],slottime[7]);

break;

case 9 :

strcpy(x->time[k],slottime[8]);

break;

case 10 :

strcpy(x->time[k],slottime[9]);

break;

case 11 :

strcpy(x->time[k],slottime[10]);

break;

case 12 :

strcpy(x->time[k],slottime[11]);

break;

}

++k7;

break;

}

}

else

{

switch(doption)

{

case 1 : doctor\_schedule11[k11]=1;

switch(k11)

{

case 1 :

strcpy(x->time[k],slottime1[0]);

break;

case 2 :

strcpy(x->time[k],slottime1[1]);

break;

case 3 :

strcpy(x->time[k],slottime1[2]);

break;

case 4 :

strcpy(x->time[k],slottime1[3]);

break;

case 5 :

strcpy(x->time[k],slottime1[4]);

break;

case 6 :

strcpy(x->time[k],slottime1[5]);

break;

case 7 :

strcpy(x->time[k],slottime1[6]);

break;

case 8 :

strcpy(x->time[k],slottime1[7]);

break;

case 9 :

strcpy(x->time[k],slottime1[8]);

break;

case 10 :

strcpy(x->time[k],slottime1[9]);

break;

case 11 :

strcpy(x->time[k],slottime1[10]);

break;

case 12 :

strcpy(x->time[k],slottime1[11]);

break;

}

++k11;

break;

case 2 : doctor\_schedule21[k2]=1;

switch(k21)

{

case 1 :

strcpy(x->time[k],slottime1[0]);

break;

case 2 :

strcpy(x->time[k],slottime1[1]);

break;

case 3 :

strcpy(x->time[k],slottime1[2]);

break;

case 4 :

strcpy(x->time[k],slottime1[3]);

break;

case 5 :

strcpy(x->time[k],slottime1[4]);

break;

case 6 :

strcpy(x->time[k],slottime1[5]);

break;

case 7 :

strcpy(x->time[k],slottime1[6]);

break;

case 8 :

strcpy(x->time[k],slottime1[7]);

break;

case 9 :

strcpy(x->time[k],slottime1[8]);

break;

case 10 :

strcpy(x->time[k],slottime1[9]);

break;

case 11 :

strcpy(x->time[k],slottime1[10]);

break;

case 12 :

strcpy(x->time[k],slottime1[11]);

break;

}

++k21;

break;

case 3 : doctor\_schedule31[k3]=1;

switch(k31)

{

case 1 :

strcpy(x->time[k],slottime1[0]);

break;

case 2 :

strcpy(x->time[k],slottime1[1]);

break;

case 3 :

strcpy(x->time[k],slottime1[2]);

break;

case 4 :

strcpy(x->time[k],slottime1[3]);

break;

case 5 :

strcpy(x->time[k],slottime1[4]);

break;

case 6 :

strcpy(x->time[k],slottime1[5]);

break;

case 7 :

strcpy(x->time[k],slottime1[6]);

break;

case 8 :

strcpy(x->time[k],slottime1[7]);

break;

case 9 :

strcpy(x->time[k],slottime1[8]);

break;

case 10 :

strcpy(x->time[k],slottime1[9]);

break;

case 11 :

strcpy(x->time[k],slottime1[10]);

break;

case 12 :

strcpy(x->time[k],slottime1[11]);

break;

}

++k31;

break;

case 4 : doctor\_schedule41[k4]=1;

switch(k41)

{

case 1 :

strcpy(x->time[k],slottime1[0]);

break;

case 2 :

strcpy(x->time[k],slottime1[1]);

break;

case 3 :

strcpy(x->time[k],slottime1[2]);

break;

case 4 :

strcpy(x->time[k],slottime1[3]);

break;

case 5 :

strcpy(x->time[k],slottime1[4]);

break;

case 6 :

strcpy(x->time[k],slottime1[5]);

break;

case 7 :

strcpy(x->time[k],slottime1[6]);

break;

case 8 :

strcpy(x->time[k],slottime1[7]);

break;

case 9 :

strcpy(x->time[k],slottime1[8]);

break;

case 10 :

strcpy(x->time[k],slottime1[9]);

break;

case 11 :

strcpy(x->time[k],slottime1[10]);

break;

case 12 :

strcpy(x->time[k],slottime1[11]);

break;

}

++k41;

break;

case 5 : doctor\_schedule51[k5]=1;

switch(k51)

{

case 1 :

strcpy(x->time[k],slottime1[0]);

break;

case 2 :

strcpy(x->time[k],slottime1[1]);

break;

case 3 :

strcpy(x->time[k],slottime1[2]);

break;

case 4 :

strcpy(x->time[k],slottime1[3]);

break;

case 5 :

strcpy(x->time[k],slottime1[4]);

break;

case 6 :

strcpy(x->time[k],slottime1[5]);

break;

case 7 :

strcpy(x->time[k],slottime1[6]);

break;

case 8 :

strcpy(x->time[k],slottime1[7]);

break;

case 9 :

strcpy(x->time[k],slottime1[8]);

break;

case 10 :

strcpy(x->time[k],slottime1[9]);

break;

case 11 :

strcpy(x->time[k],slottime1[10]);

break;

case 12 :

strcpy(x->time[k],slottime1[11]);

break;

}

++k51;

break;

case 6 : doctor\_schedule61[k6]=1;

switch(k61)

{

case 1 :

strcpy(x->time[k],slottime1[0]);

break;

case 2 :

strcpy(x->time[k],slottime1[1]);

break;

case 3 :

strcpy(x->time[k],slottime1[2]);

break;

case 4 :

strcpy(x->time[k],slottime1[3]);

break;

case 5 :

strcpy(x->time[k],slottime1[4]);

break;

case 6 :

strcpy(x->time[k],slottime1[5]);

break;

case 7 :

strcpy(x->time[k],slottime1[6]);

break;

case 8 :

strcpy(x->time[k],slottime1[7]);

break;

case 9 :

strcpy(x->time[k],slottime1[8]);

break;

case 10 :

strcpy(x->time[k],slottime1[9]);

break;

case 11 :

strcpy(x->time[k],slottime1[10]);

break;

case 12 :

strcpy(x->time[k],slottime1[11]);

break;

}

++k61;

break;

case 7 : doctor\_schedule71[k7]=1;

switch(k71)

{

case 1 :

strcpy(x->time[k],slottime1[0]);

break;

case 2 :

strcpy(x->time[k],slottime1[1]);

break;

case 3 :

strcpy(x->time[k],slottime1[2]);

break;

case 4 :

strcpy(x->time[k],slottime1[3]);

break;

case 5 :

strcpy(x->time[k],slottime1[4]);

break;

case 6 :

strcpy(x->time[k],slottime1[5]);

break;

case 7 :

strcpy(x->time[k],slottime1[6]);

break;

case 8 :

strcpy(x->time[k],slottime1[7]);

break;

case 9 :

strcpy(x->time[k],slottime1[8]);

break;

case 10 :

strcpy(x->time[k],slottime1[9]);

break;

case 11 :

strcpy(x->time[k],slottime1[10]);

break;

case 12 :

strcpy(x->time[k],slottime1[11]);

break;

}

++k71;

break;

}

}

cout<<"\n Dear Patient;\n Your appointment is booked\n Doctor name :"<<x->doctornam[k]<<"\n Date :"<<x->date[k]<<"\ "<<x->month[k]<<"\ "<<x->year[k]<<"\n Time :";

cout<<x->time[k];

cout<<"\n Kindly be on time";

}

void book\_appointment(int n1,int tnum)

{

node a;

if(first[n1]==nullptr)

{

first[n1] = new (struct patient);

fill\_details(first[n1],n1);

first[n1]->next=nullptr;

s[n1].incr\_task();

}

else

{

a=first[n1];

while(a->next!=nullptr)

{

a=a->next;

}

a->next= new(struct patient);

a=a->next;

fill\_details(a,n1);

s[n1].incr\_task();

}

}

void show\_appointment(int n)

{

node a;

int i=0;

a=first[n];

cout<<"\n Appointment No. \t Doctor name \t Date \t time";

while(a!=nullptr)

{

cout<<"\n"<<i+1<<"\t"<<a->doctornam[i]<<"\t"<<a->date[i]<<"/"<<a->month[i]<<"/"<<a->year[i]<<"\t"<<a->time[i];

a=a->next;

++i;

}

}

void delete\_appointment(int n)

{

int num;

cout<<"\n Enter the appointment number you want to delete";

cin>>num;

node a,b;

a=first[n];

if(num==1 && s[n].task\_num==1)

{

delete a;

}

if(num ==1 && s[n].task\_num!=1)

{

b=a;

a=a->next;

delete b;

first[n]=a;

}

else

{

while(num>1)

{

b=a;

a=a->next;

}

b->next=a->next;

delete a;

}

}

int search(char user1[15])

{

int i,count=0;

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

{

if(strcmp(user1,s[i].username)==0)

{

++count;

return 0;

}

}

if(count==0)

{

return -1;

}

count=0;

return 0;

}

void reg::add\_details()

{

int ret,x;

++n;

char user1[15];

cout<<"\n Enter your name";

cin>>name;

cout<<"\n Enter your gmail";

cin>>gmail;

cout<<"\n Enter your phone number";

cin>>phone;

do

{

cout<<"\n Enter your username (maximum character 15)";

cin>>user1;

ret=search(user1);

if(ret==0)

{

cout<<"\n please choose other username.someone has already taken it";

}

else

{

strcpy(username,user1);

}

}while(ret==0);

cout<<"\n Enter your password (maximum character 15)";

cin>>password;

cout<<"\n which security question do you want to keep (maximum character 15)? \nPress 1 : What is the name of your first school?\n Press 2 : What is the name of your pet?";

do

{

cin>>x;

switch(x)

{

case 1 : cin>>ans;

que=x;

break;

case 2 :cin>>ans;

que=x;

break;

default: cout<<"\n Oops!! Wrong input given \n please try again";

}

}while(x!=1 && x!=2);

}

void forgot()

{

char g1[30],p1[10];

char a1[20];

int i=0,count=0;

cout<<"\n Enter your gmail";

cin>>g1;

cout<<"\n Enter your phone number";

cin>>p1;

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

{

if(strcmp(g1,s[i].gmail)==0 && strcmp(p1,s[i].phone)==0)

{

if(s[i].que == 1)

{

cout<<"\n What is the name of your first school?";

cin>>a1;

}

else

{

cout<<"\n What is the name of your pet?";

cin>>a1;

}

if(strcmp(a1,s[i].ans)==0)

{

cout<<"\n Your username and password are";

cout<<s[i].username<<"\t"<<s[i].password;

}

else

{

cout<<"\n You had given wrong answer";

}

++count;

}

}

if(count==0)

{

cout<<"Your entered gmail and phone number didn't matched";

}

count=0;

}

int search1(char u1[15],char p1[15])

{

int i,count=0;

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

{

if(strcmp(u1,s[i].username)==0 && strcmp(p1,s[i].password)==0)

{

++count;

return i;

}

}

if(count==0)

{

return -1;

}

return 0;

}

int dsearch1(char u1[15],char p1[15])

{

int i,count=0,t1,t2;

for(i=0;i<7;i++)

{

if(strcmp(u1,d[i].dusername)==0 && strcmp(p1,d[i].dpassword)==0)

{

++count;

return i;

}

}

if(count==0)

{

return -1;

}

return 0;

}

int login()

{

int ret;

char u1[15],p1[15];

do

{

cout<<"\n Enter your Username";

cin>>u1;

cout<<"\n Enter your password";

cin>>p1;

ret = search1(u1,p1);

if(ret!=-1 )

{

return ret;

}

else

{

cout<<"\n PLease try again";

}

}while(ret!=1);

return 0;

}

int dlogin()

{

int ret;

char u1[15],p1[15];

do

{

cout<<"\n Enter your Username";

cin>>u1;

cout<<"\n Enter your password";

cin>>p1;

ret = dsearch1(u1,p1);

if(ret!=-1 )

{

return ret;

}

else

{

cout<<"\n PLease try again";

}

}while(ret!=1);

return 0;

}

void task\_selection(int n1)

{

int x,tnum;

char ans1;

do

{

cout<<"\n Press 1 : To book appointment \n Press 2 : To check appointment history \n Press 3 : To delete appointment \n Press 4 :Medical History";

cin>>x;

switch(x)

{

case 1 :

book\_appointment(n1,1);

break;

case 2 :show\_appointment(n1);

break;

case 3 :delete\_appointment(n1);

cout<<"\n The appointment has been deleted";

break;

default :cout<<"\n Wrong input given";

}

cout<<"\n Do you want to continue(y/n)?";

cin>>ans1;

}while(ans1=='y');

}

void dtask\_selection(int n1)

{

int x,tnum;

char ans1;

do

{

cout<<"\n Press 1 : To check appointment \n Press 2 : To fill patient medical history\n ";

cin>>x;

switch(x)

{

case 1 :check\_appointment(n1);

break;

default :cout<<"\n Wrong input given";

}

cout<<"\n Do you want to continue(y/n)?";

cin>>ans1;

}while(ans1=='y');

}

int main()

{

char ans1;

int x,ret;

doctor\_details();

do

{

cout<<"\n Press 1 : To register \n Press 2 : Patient login \n Press 3 :Doctor login \n Press 4 : forgot username/password \n Enter your choice";

cin>>x;

switch(x)

{

case 1 : s[n].add\_details();

break;

case 2 : ret=login();

if(ret!=-1)

{

task\_selection(ret);

}

break;

case 3 : ret = dlogin();

if(ret!=-1)

{

dtask\_selection(ret);

}

break;

case 4 :forgot();

break;

default :cout<<"\n Wrong input given!!";

}

cout<<"\n Do you want to continue(y/n)?";

cin>>ans1;

}while(ans1=='y');

}