```cpp
#include<iostream>
#include<string>
using namespace std;

// Hospital Management System
// 12300453 , مهند ماجد عبد العظيم محمد , Class 7
// 12300467 , كمال عاطف كمال , Class 7
// 12300396 , معاذ هشام , Class 6

class node{
    public:
    int id ;
    string name ;
    string Specialization ;
    string hospitalbranch ;
    string appointmentschedule ;
    node*next ;
};


class DoctorLinkedList {
    public:
    node*head;


    // add(done) , remove() , display appointment(done) //

    DoctorLinkedList()
    {
        head = NULL ;
    }

    bool IsEmpty()
    {
        if (head==NULL)
        {
            return true ;
        }
        else
        {
            return false ;
        }
    }

    void Display()
    {
        node*temp= head ;
        while(temp!= NULL)
        {
            cout<<"DOCTOR name :"<<temp->name <<endl;
            cout<<"DOCTOR id :"<<temp->id <<endl;
            cout<<"DOCTOR Specialization :"<<temp->Specialization <<endl;
            cout<<"DOCTOR hospitalbranch :"<<temp->hospitalbranch <<endl;
            temp = temp->next ;
        }
        cout<<endl;
    }

    bool Search(int id)
    {
        bool key = false;
        node*temp= head ;

        while(temp!= NULL)
        {
            if(temp->id== id)
            {
               key = true ;
               break;
            }

            temp = temp->next ;
        }

        return key ;
```

```
}

void AddFirst(int id , string branch ,string name, string Specialization)
{
    node* newnode = new node();
    newnode->hospitalbranch = branch ;
    newnode->id=id ;
    newnode->name=name ;
    newnode->Specialization=Specialization ;

    if(IsEmpty())
    {
        newnode->next = NULL ;
        head= newnode ;
    }
    else
    {
        newnode->next = head ;
        head = newnode ;
    }
}

void AddtBefore(int beforeid , int id , string branch  ,string name, string Specialization)
{
    if(IsEmpty())
        AddFirst(id , branch , name , Specialization) ;
    if(Search(beforeid))
    {
        node*newnode = new node() ;
        newnode->id = id ;
        newnode->hospitalbranch = branch ;
        newnode->name = name ;
        newnode->Specialization = Specialization ;

        node*temp = head ;

        while (temp !=NULL && temp->next->id !=id)
        {
            temp= temp->next ;
        }
        newnode->next=temp->next ;
        temp->next=newnode;
    }
    else
    cout<<"SORRY, THE ITEM WAS NOT FOUND "<<endl;
}

void Append(int id , string branch ,string name, string Specialization)
{
    if (IsEmpty())
        AddFirst(id ,branch , name , Specialization) ;
    else
    {
        node*temp = head ;
            while (temp->next != NULL)
            {
                temp = temp->next ;
            }
        node* newnode = new node() ;

        newnode->id = id ;
        newnode->hospitalbranch = branch ;
        newnode->name = name ;
        newnode->Specialization = Specialization ;

        temp->next = newnode ;
        newnode->next = NULL ;
    }
}

void Remove(int id)
{
    node*delpointer = head ;

    if( head->id == id)
```

```
        {
            head = head->next ;
            delete delpointer ;
        }
        else
        {
            node*prevpointer = NULL ;
            while(delpointer->id != id)
            {
                prevpointer = delpointer ;
                delpointer = delpointer->next ;
            }
            prevpointer->next = delpointer->next ;
            delete delpointer ;
        }
    }

    void AddAppointment(int id, string appointmentschedule)
    {
        if (Search(id))
        {
            node* temp = head;
            while (temp != NULL && temp->id != id)
            {
                temp = temp->next;
            }

            temp->appointmentschedule=appointmentschedule;
        }
        else
        {
            cout << "Doctor not found. Cannot add appointment." << endl;
        }
    }

    void DisplayAppointment()
    {
        node*temp= head ;
        while(temp!= NULL)
        {
            cout<<"AVAILABLE APPOINTMENTS :"<<temp->appointmentschedule <<endl;
            temp = temp->next ;
        }
        cout<<endl;
    }


};


class nodep{
    public:
    int id ;
    string name ;
    string address ;
    string appointment;
    nodep*next ;
};

class PatientsLinkedList {
    public:
    nodep*head;

   // add(done) , remove(done) , book(done) ,search (done) , update(done) //

    PatientsLinkedList()
    {
        head = NULL ;
    }

    bool IsEmpty()
    {
        if (head==NULL)
        {
            return true ;
```

```
    }
    else
    {
        return false ;
    }
}

bool Search(int id)
{
    bool key = false;
    nodep*temp= head ;

    while(temp!= NULL)
    {
        if(temp->id== id)
        {
           key = true ;
           break;
        }

        temp = temp->next ;
    }

    return key ;
}

void InsertFirst(int id ,string name , string address , string appointment)
{
    nodep* newnodep = new nodep();
    newnodep->id= id ;
    newnodep->name= name ;
    newnodep->address = address ;
    newnodep->appointment= appointment ;

    if (IsEmpty())

    {
        newnodep->next = NULL ;
        head= newnodep ;
    }
    else
    {
        newnodep->next =head;
        head = newnodep ;
    }
}

void AddtBefore(int beforeid , int id ,string name, string address , string appointment)
{
    if(IsEmpty())
        InsertFirst(id , name , address, appointment) ;
    if(Search(beforeid))
    {
        nodep* newnodep = new nodep();
        newnodep->id = id ;
        newnodep->name = name ;
        newnodep->address = address ;
        newnodep->appointment = appointment ;

        nodep*temp = head ;

        while (temp !=NULL && temp->next->id !=id)
        {
            temp= temp->next ;
        }
        newnodep->next=temp->next ;
        temp->next=newnodep;
    }
    else
    cout<<"SORRY, THE ITEM WAS NOT FOUND "<<endl;
}

void Append( int id ,string name, string address , string appointment)
{
    if (IsEmpty())
```

```
            InsertFirst(id , name , address, appointment) ;
        else
        {
            nodep*temp = head ;
                while (temp->next != NULL)
                {
                    temp = temp->next ;
                }
            nodep* newnodep = new nodep();
            newnodep->id = id ;
            newnodep->name = name ;
            newnodep->address = address ;
            newnodep->appointment = appointment ;
            temp->next = newnodep ;
            newnodep->next = NULL ;
        }
    }

    void Display()
    {
        nodep*temp= head ;
        while(temp!= NULL)
        {
            cout<<"PATIENT name :"<<temp->name <<endl;
            cout<<"PATIENT id :"<<temp->id <<endl;
            cout<<"PATIENT address :"<<temp->address <<endl;
            cout<<"PATIENT appointment :"<<temp->appointment <<endl;
            temp = temp->next ;
        }
        cout<<endl;
    }

    void Remove(int id)
    {
        nodep*delpointer = head ;

        if( head->id == id)
        {
            head = head->next ;
            delete delpointer ;
        }
        else
        {
            nodep*prevpointer = NULL ;
            while(delpointer->id != id)
            {
                prevpointer = delpointer ;
                delpointer = delpointer->next ;
            }
            prevpointer->next = delpointer->next ;
            delete delpointer ;
        }
    }

    bool SearchByName(string name)
    {
        bool key = false;
        nodep*temp= head ;

        while(temp!= NULL)
        {
            if(temp->name== name)
            {
               key = true ;
               break;
            }
            temp = temp->next ;
        }

        return key ;
    }

    bool BookAppointment(string appointment)
    {
        bool key = false;
```

```
        nodep*temp= head ;

        while(temp!= NULL)
        {
            if(temp->appointment == appointment)
            {
               key = true ;
               break;
            }
            temp = temp->next ;
        }
        return key ;
    }

    void Update(int old_ID ,int id ,string name , string address , string appointment)
    {
        if (Search(old_ID))
        {
            nodep*temp = head ;
            nodep*deleteditem ;
            while (temp->id != old_ID)
            {
                temp = temp->next ;
            }
            deleteditem->next = temp->next ;
            temp->id = id ;
            temp->address = address ;
            temp->appointment = appointment ;
            temp->name = name ;
            delete deleteditem ;
        }
        else
        cout<<"SORRY, ITEM WAS NOT FOUND"<<endl;
    }


};


// branch hospital
class Branch {
public:
    int ID;
    string name;
    string location;
    string appointment;
    Branch* next;

    Branch() {
        next = NULL;
        ID = 0;
    }
};

class HospitalBranchLinkedList {
public:
    Branch* head;

    HospitalBranchLinkedList()
    {
        head = NULL;
    }

    bool isEmpty()
    {
        return head == NULL;
    }

    void insertBranchFirst(string name, string location, int ID, string appointment)
    {
        Branch* newBranch = new Branch();
        newBranch->name = name;
        newBranch->location = location;
        newBranch->ID = ID;
        newBranch->appointment= appointment;
```

```
    if (isEmpty()) {
        newBranch->next = NULL;
        head = newBranch;
    } else {
        newBranch->next = head;
        head = newBranch;
    }
    cout << "Branch added at first" << endl;
}

void insertBranchBefore(int beforeID, int ID, string name, string location, string appointment)
{
    if (isEmpty()) {
        insertBranchFirst(name, location, ID , appointment);
        return;
    }

    Branch* newBranch = new Branch();
    newBranch->ID = ID;
    newBranch->name = name;
    newBranch->location = location;
    newBranch->appointment= appointment;

    Branch* temp = head;
    Branch* prev = nullptr;

    while (temp != nullptr && temp->ID != beforeID)
    {
        prev = temp;
        temp = temp->next;
    }

    if (temp == nullptr)
    {
        cout << "SORRY, THE ITEM WAS NOT FOUND" << endl;
        delete newBranch;
    }

    else
    {
        if(prev == nullptr)
        {
            newBranch->next = head;
            head = newBranch;
        }

        else
        {
            newBranch->next = temp;
        prev->next = newBranch;
        }
        cout << "Branch added before ID: " << beforeID << endl;
    }
}


    void Append( int id ,string name , string location , string appointment)
{
    if (isEmpty())
        insertBranchFirst(name,location,id,appointment) ;
    else
    {
        Branch*temp = head ;
            while (temp->next != NULL)
            {
                temp = temp->next ;
            }
        Branch* newnodep = new Branch();
        newnodep->ID = id ;
        newnodep->name = name ;
        newnodep->location = location ;
        newnodep->appointment= appointment;

        temp->next = newnodep ;
```

```
            newnodep->next = NULL ;
        }
    }

    void removeBranch(int branchID)
    {
        Branch* current = head;
        Branch* prev = nullptr;

        while (current != nullptr && current->ID != branchID)
        {
            prev = current;
            current = current->next;
        }

        if (current == nullptr)
        {
            cout << "Branch with ID " << branchID << " not found." << endl;
            return;
        }

        if (prev == nullptr)
        {
            head = current->next;
        }

        else
        {
            prev->next = current->next;
        }
        delete current;
        cout << "Branch with ID " << branchID << " removed." << endl;
    }

        void displayBranchData()
    {
        Branch* temp = head;
        while (temp != NULL)
        {
            cout << "Name: " << temp->name << endl;
            cout << "Location: " << temp->location << endl;
            cout << "ID: " << temp->ID << endl;
            cout << "APPOINTMENT: " << temp->appointment << endl;
            temp = temp->next;
        }
        cout << "End of branches" << endl;
    }

    bool searchBranch(int searchID)
    {
        Branch* temp = head;
        while (temp != NULL)
        {
            if (temp->ID == searchID)
            return true;
            temp = temp->next;
        }
        return false;
    }

    void DisplayBranchesAppointments()
    {
        Branch*temp= head ;
        while(temp!= NULL)
        {
            cout<<"PATIENT appointment :"<<temp->appointment <<endl;
            temp = temp->next ;
        }
        cout<<endl;
    }


};
```

```
int main()
{
    DoctorLinkedList DList ;
    PatientsLinkedList PList ;
    HospitalBranchLinkedList BList ;


    int id ;
    string name ;
    string Specialization ;
    string hospitalbranch ;
    string appointmentschedule ;

    cout<<"ENTER DOCTOR NAME :"<<endl;
    cin>>name ;
    cout<<"ENTER DOCTOR ID :"<<endl;
    cin>>id ;
    cout<<"ENTER DOCTOR Specialization :"<<endl;
    cin>>Specialization;
    cout<<"ENTER DOCTOR hospitalbranch :"<<endl;
    cin>>hospitalbranch;
    cout<<"ENTER APPOINTMENT SCHEDULE :"<<endl;
    cin>>appointmentschedule;
    DList.AddFirst(id ,hospitalbranch ,name , Specialization);
    DList.AddAppointment(id,appointmentschedule);
    DList.Display();
    DList.DisplayAppointment();

    cout<<endl;

    cout<<"ENTER DOCTOR NAME :"<<endl;
    cin>>name ;
    cout<<"ENTER DOCTOR ID :"<<endl;
    cin>>id ;
    cout<<"ENTER DOCTOR Specialization :"<<endl;
    cin>>Specialization;
    cout<<"ENTER DOCTOR hospitalbranch :"<<endl;
    cin>>hospitalbranch;
    cout<<"ENTER APPOINTMENT SCHEDULE :"<<endl;
    cin>>appointmentschedule;
    DList.AddFirst(id ,hospitalbranch ,name , Specialization);
    DList.AddAppointment(id,appointmentschedule);
    DList.Display();
    DList.DisplayAppointment();

    cout<<endl;

    cout<<"ENTER ID OF DOCTOR YOU WANT TO REMOVE :"<<endl;
    cin>>id;
    DList.Remove(id);
    DList.Display();









    string address ;
    string appointment ;
    int old_ID ;
    cout<<"ENTER PATIENT NAME :"<<endl;
    cin>>name ;
    cout<<"ENTER PATIENT ID :"<<endl;
    cin>>id ;
    cout<<"ENTER PATIENT ADDRESS :"<<endl;
    cin>>address;
    cout<<"ENTER APPOINTMENT :"<<endl;
    cin>>appointment;
    PList.InsertFirst(id , name , address , appointment);
```

```
PList.Display();
cout<<endl;
if(PList.BookAppointment(appointment))
cout<<"YOU HAVE BOOKED AN APPOINTMENT SUCCESSFULLY ."<<endl;
else
cout<<"THE APPOINTMENT WAS NOT BOOKED ."<<endl;
cout<<endl;


cout<<"ENTER PATIENT NAME :"<<endl;
cin>>name ;
cout<<"ENTER PATIENT ID :"<<endl;
cin>>id ;
cout<<"ENTER PATIENT ADDRESS :"<<endl;
cin>>address;
cout<<"ENTER APPOINTMENT :"<<endl;
cin>>appointment;
PList.InsertFirst(id , name , address , appointment);
PList.Display();
cout<<endl;
if(PList.BookAppointment(appointment))
cout<<"YOU HAVE BOOKED AN APPOINTMENT SUCCESSFULLY ."<<endl;
else
cout<<"THE APPOINTMENT WAS NOT BOOKED ."<<endl;
cout<<endl;


cout<<"ENTER THE PATIENT YOU WANT TO REMOVE :";
cin>>id;
PList.Remove(id);
PList.Display();
cout<<endl;


cout<<"ENTER NAME OF THE PATIENT YOU WANT TO SEARCH ABOUT :";
cin>>name;
PList.SearchByName(name);
if(PList.SearchByName(name))
cout<<"THE PATIENT IS EXIST ."<<endl;
else
cout<<"THE PATIENT IS NOT EXIST ."<<endl;
cout<<endl;


cout<<"ENTER THE ID OF THE PATIENT YOU WANT TO UPDATE HIS INFORMATION :";
cin>>old_ID;
cout<<"ENTER PATIENT'S NEW NAME :"<<endl;
cin>>name ;
cout<<"ENTER PATIENT'S NEW ID :"<<endl;
cin>>id ;
cout<<"ENTER PATIENT'S NEW ADDRESS :"<<endl;
cin>>address;
cout<<"ENTER THE NEW APPOINTMENT :"<<endl;
cin>>appointment;
PList.Update(old_ID , id , name , address , appointment );
PList.Display();









string location;
cout<<"ENTER BRANCH NAME :"<<endl;
cin>>name ;
cout<<"ENTER BRANCH ID :"<<endl;
cin>>id ;
cout<<"ENTER BRANCH LOCATION :"<<endl;
cin>>location ;
cout<<"ENTER BRANCH'S APPOINTMENT :"<<endl;
```

```
        cin>>appointment ;
        BList.insertBranchFirst(name , location , id , appointment);
        BList.displayBranchData();
        cout<<endl;

        cout<<"ENTER BRANCH NAME :"<<endl;
        cin>>name ;
        cout<<"ENTER BRANCH ID :"<<endl;
        cin>>id ;
        cout<<"ENTER BRANCH LOCATION :"<<endl;
        cin>>location ;
        cout<<"ENTER BRANCH'S APPOINTMENT :"<<endl;
        cin>>appointment ;
        BList.insertBranchFirst(name , location , id , appointment);
        BList.displayBranchData();
        cout<<endl;



        cout<<"ENTER ID OF THE BRANCH YOU WANT TO SEARCH ABOUT :";
        cin>>id;
        BList.searchBranch(id);

        cout<<"ENTER ID OF THE BRANCH YOU WANT TO REMOVE :";
        cin>>id;
        BList.removeBranch(id);

        cout<<"HERE WE WILL DISPLAY ALL BRANCHES APPOINTMENTS :"<<endl;
        BList.DisplayBranchesAppointments();


        return 0 ;
    }
```