pragma solidity ^0.5.1;

pragma experimental ABIEncoderV2;

contract Hospital\_Trainee\_Competency {

address public \_NationalAuthority;

struct Hospital {

address \_HospitalAddress;

uint \_HospitalID;

string \_HospitalName;

}

Hospital H;

mapping (address => Hospital) Hos;

Hospital[] ListOfHospital;

struct Trainee {

address \_TraineeAddress;

uint \_TraineeID;

string \_TraineeName;

string \_TrHospitaName;

string \_Category;

bytes32 \_TraineeCertificate;

}

Trainee T;

mapping (address => Trainee) Tra;

Trainee[] ListOfTrainee;

address[] public OfficialEmployees;

event delete\_Hospital(address \_HospitalAddress,uint \_HospitalID);

event add\_Records(address \_HospitalAddress, address \_TraineeAddress, uint \_TraineeID, string \_TraineeName, string \_TrHospitaName, string \_Category, bytes32 \_TraineeCertificate);

constructor () public {

\_NationalAuthority = msg.sender;

}

function Authorise\_Hospital(address \_HospitalAddress, uint \_HospitalID, string memory \_HospitalName) public {

require(msg.sender == \_NationalAuthority);

Hospital storage hos = Hos[\_HospitalAddress];

require(\_HospitalAddress != hos.\_HospitalAddress);

require(\_HospitalID != hos.\_HospitalID);

hos.\_HospitalAddress = \_HospitalAddress;

hos.\_HospitalID = \_HospitalID;

hos.\_HospitalName = \_HospitalName;

ListOfHospital.push(hos);

}

function getListOfHospitals() public view returns(Hospital[] memory){

return ListOfHospital;

}

function See\_Authorised\_Hospital(address \_HospitalAddress) view public returns (address, uint, string memory){

return (Hos[\_HospitalAddress].\_HospitalAddress, Hos[\_HospitalAddress].\_HospitalID, Hos[\_HospitalAddress].\_HospitalName);

}

function Add\_Employees(address \_HospitalAddress, address \_TraineeAddress) public {

require(Hos[\_HospitalAddress].\_HospitalAddress == msg.sender);

OfficialEmployees.push(\_TraineeAddress);

}

function getListOfOfficialEmployees() view public returns (address[] memory){

return (OfficialEmployees);

}

function Add\_Records(address \_HospitalAddress, address \_TraineeAddress, uint \_TraineeID, string memory \_TraineeName, string memory \_TrHospitaName, string memory \_Category, bytes32 \_TraineeCertificate)public {

require(Hos[\_HospitalAddress].\_HospitalAddress == msg.sender);

for(uint i = 0; i <= OfficialEmployees.length; i++)

{

if(OfficialEmployees[i] == \_TraineeAddress)

{

Trainee storage tra = Tra[\_TraineeAddress];

tra.\_TraineeAddress = \_TraineeAddress;

tra.\_TraineeID = \_TraineeID;

tra.\_TraineeName = \_TraineeName;

tra.\_TrHospitaName = \_TrHospitaName;

tra.\_Category = \_Category;

tra.\_TraineeCertificate = \_TraineeCertificate;

ListOfTrainee.push(tra);

emit add\_Records(\_HospitalAddress, \_TraineeAddress, \_TraineeID, \_TraineeName, \_TrHospitaName, \_Category, \_TraineeCertificate);

break;

}

}

}

function getListOfTraineesRecords() public view returns(Trainee[] memory) {

return ListOfTrainee;

}

function deleteHospital(address \_HospitalAddress, uint \_HospitalID) public {

require(msg.sender == \_NationalAuthority);

for(uint i = 0; i < ListOfHospital.length; i++)

{

if(\_HospitalAddress == ListOfHospital[i].\_HospitalAddress && \_HospitalID == ListOfHospital[i].\_HospitalID)

{

delete ListOfHospital[i];

emit delete\_Hospital(\_HospitalAddress, \_HospitalID);

break;

}

}

}

function deleteEmployee(address \_HospitalAddress, address \_TraineeAddress, uint \_TraineeID) public {

require(msg.sender == Hos[\_HospitalAddress].\_HospitalAddress);

}

}