CHAT HISTORY

```
#include <iostream>
#include <stack>
using namespace std;

class ChatStack {
    stack<string> chats;

public:
    void sendMessage(string msg) {
        chats.push(msg);
    }

    void deleteLast() {
        if (!chats.empty()) {
            chats.pop();
        }
    }

    void showHistory() {
        stack<string> temp = chats;
        cout << "Chat History:\n";
        while (!temp.empty()) {
            cout << temp.top() << endl;
```

```
            temp.pop();
        }
    }
};
```

Doctor list

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

class DoctorNode {
public:
    string name;
    DoctorNode* next;

    DoctorNode(string n) {
        name = n;
        next = NULL;
    }
};

class DoctorList {
    DoctorNode* head;

public:
    DoctorList() {
        head = NULL;
    }
```

```
    void addDoctor(string name) {
        DoctorNode* newNode = new DoctorNode(name);
        newNode->next = head;
        head = newNode;
    }

    void displayDoctors() {
        DoctorNode* temp = head;
        cout << "Doctors List:\n";
        while (temp) {
            cout << temp->name << endl;
            temp = temp->next;
        }
    }
};
```

Hospital Tree

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

class TreeNode {
public:
    string name;
    TreeNode* left;
    TreeNode* right;

    TreeNode(string n) {
        name = n;
        left = right = NULL;
    }
};

class HospitalTree {
public:
    TreeNode* root;

    HospitalTree() {
        root = new TreeNode("Hospital");
```

```
        root->left = new TreeNode("Cardiology");
        root->right = new TreeNode("Neurology");
    }

    void display(TreeNode* node, int level = 0) {
        if (node) {
            cout << string(level * 2, ' ') << node->name << endl;
            display(node->left, level + 1);
            display(node->right, level + 1);
        }
    }
};
```

Main File

```
#include "patient_queue.h"
#include "chat_stack.h"
#include "doctor_linkedlist.h"
#include "hospital_tree.h"
#include "referral_graph.h"
#include "auth_hashing.h"

int main() {

    // Queue
    PatientQueue pq;
    pq.addPatient("Ravi");
    pq.addPatient("Anita");
    pq.servePatient();

    cout << endl;

    // Stack
    ChatStack chat;
    chat.sendMessage("Hello Doctor");
    chat.sendMessage("Need Appointment");
```

```
chat.showHistory();

cout << endl;

// Linked List
DoctorList dl;
dl.addDoctor("Dr. Smith");
dl.addDoctor("Dr. John");
dl.displayDoctors();

cout << endl;

// Tree
HospitalTree ht;
ht.display(ht.root);

cout << endl;

// Graph
ReferralGraph rg;
rg.addReferral("Dr. Smith", "Dr. John");
rg.showReferrals();
```

```
    cout << endl;

    // Hashing
    AuthSystem auth;
    auth.registerUser("patient1", "1234");

    cout << "Login Success: "
        << (auth.login("patient1", "1234") ? "Yes" : "No") << endl;

    return 0;
}
```

Patient Queue

```
#include <iostream>
#include <queue>
using namespace std;

class PatientQueue {
    queue<string> q;

public:
    void addPatient(string name) {
        q.push(name);
        cout << "Patient added: " << name << endl;
    }

    void servePatient() {
        if (!q.empty()) {
            cout << "Serving patient: " << q.front() << endl;
            q.pop();
        } else {
            cout << "No patients in queue\n";
        }
    }
```

```
};
```

README.MD

Hospital Management Mini Project (C++)

Data Structures Used

1. Queue – Patient Queue
2. Stack – Chat History
3. Linked List – Doctor List
4. Tree – Hospital Departments
5. Graph – Referral System
6. Hashing – User Authentication

Language

C++

How to Run

```bash
g++ main.cpp -o hospital
./hospital
```

Referral Graph

```
#include <iostream>
#include <map>
#include <vector>
using namespace std;

class ReferralGraph {
    map<string, vector<string>> graph;

public:
    void addReferral(string from, string to) {
        graph[from].push_back(to);
    }

    void showReferrals() {
        cout << "Doctor Referrals:\n";
        for (auto pair : graph) {
            cout << pair.first << " -> ";
            for (string doc : pair.second) {
                cout << doc << " ";
```

```
            }
            cout << endl;
        }
    }
};
```

User Authentication

```
#include <iostream>
#include <unordered_map>
using namespace std;

class AuthSystem {
    unordered_map<string, int> users;

public:
    int hashPassword(string password) {
        int hash = 0;
        for (char c : password) {
            hash += c;
        }
        return hash;
    }

    void registerUser(string username, string password) {
        users[username] = hashPassword(password);
    }

    bool login(string username, string password) {
```

```
        return users[username] == hashPassword(password);
    }
};
```