NATIONAL UNIVERSITY OF MODERN LANGUAGES (NUML) ISLAMABAD
Faculty of Computing and Engineering
Department of Software Engineering

Data Structures And Algorithms

SEMESTER PROJECT

"Old Age Home Management System"

Group Members:

Students' names	Roll Numbers	System ID
Anmol Ihsan	SE-9244307	NUML-F24-13857
Syed Mujtaba Hassan	SE-9244375	NUML-F24-49218
Fatima Salehi	SE-9244551	NUML-F24-83218
Ismail Khalid	SP-23902	NUML-S24-24013

Course Instructor:

DR. MARYAM IMTIAZ MALIK

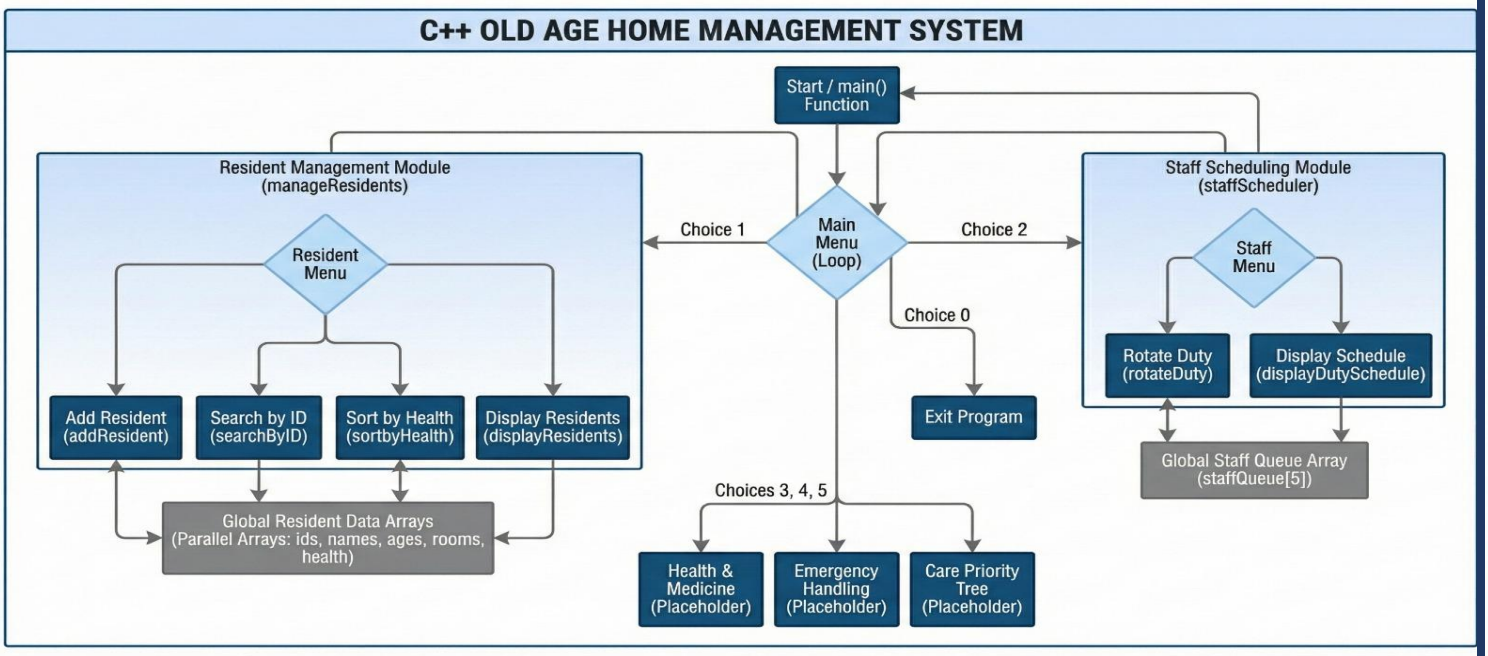
Introduction

The Old Age Home Management System is designed to streamline the management of residents, staff, health logs, and care priorities in an elder care facility. It addresses real-world needs such as elder care, emergency response, and health tracking.

System Overview

The system consists of the following modules:

- Resident Management
- Staff Scheduling
- Health & Medicine Logs
- Care Priority Tree
- Exit



Technology Used

- Language: C++
- Data Structures: Arrays, Search, Sort, Queue, Circular Queue, Linked List, Stack, Recursion, Binary Search Tree.
- IDE: Dev-C++

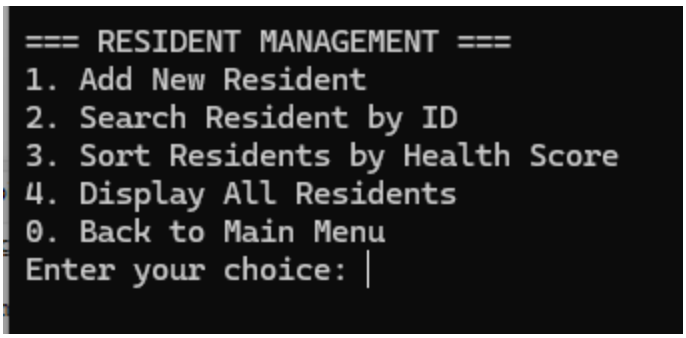
Module Descriptions

1. Resident Management:

- **Objective:** Manage resident details such as ID, name, age, room, and health score.
- Data Structure Used: Arrays
- **Key Functions:** addResident(), searchByID(), sortByHealth(), displayResidents()
- Sample Code Snippet:

```
void addResident() {
    cout << "\nEnter ID: ";
    cin >> ids[countRes];
    ...
}
```

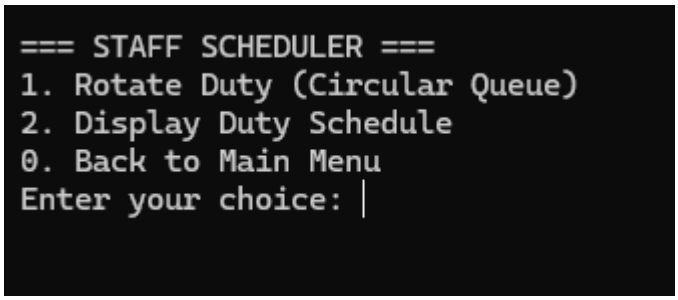
- **Sample Output Screenshot:**



2. Staff Scheduling:

- **Objective:** Rotate staff duties using a circular queue.
- **Data Structure Used:** Circular Queue (Array)
- **Key Functions:** rotateDuty(), displayDutySchedule()

- **Sample Output Screenshot:**

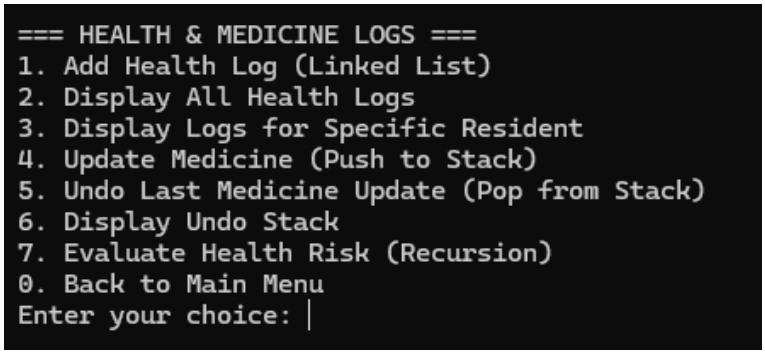


3. Health & Medicine Logs:

- **Objective:** Track health logs, manage medicine updates, and evaluate health risks.

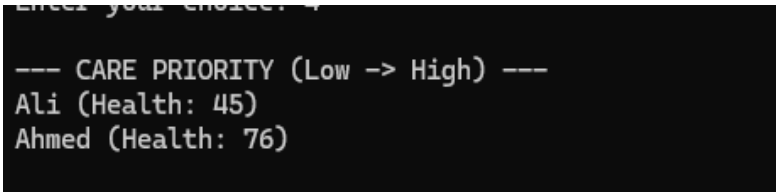
- **Data Structure Used:** Linked List, Stack, Recursion
- **Key Functions:** addHealthLog(), updateMedicine(), undoMedicineUpdate(), evaluateHealthRisk()

- **Sample Output Screenshot:**



4. Care Priority Tree:

- **Objective:** Sort residents by health score using a Binary Search Tree.
- Data Structure Used: BST
- **Key Functions:** insertTree(), inorder()



- **Sample Output Screenshot:**

5. Unique Features:

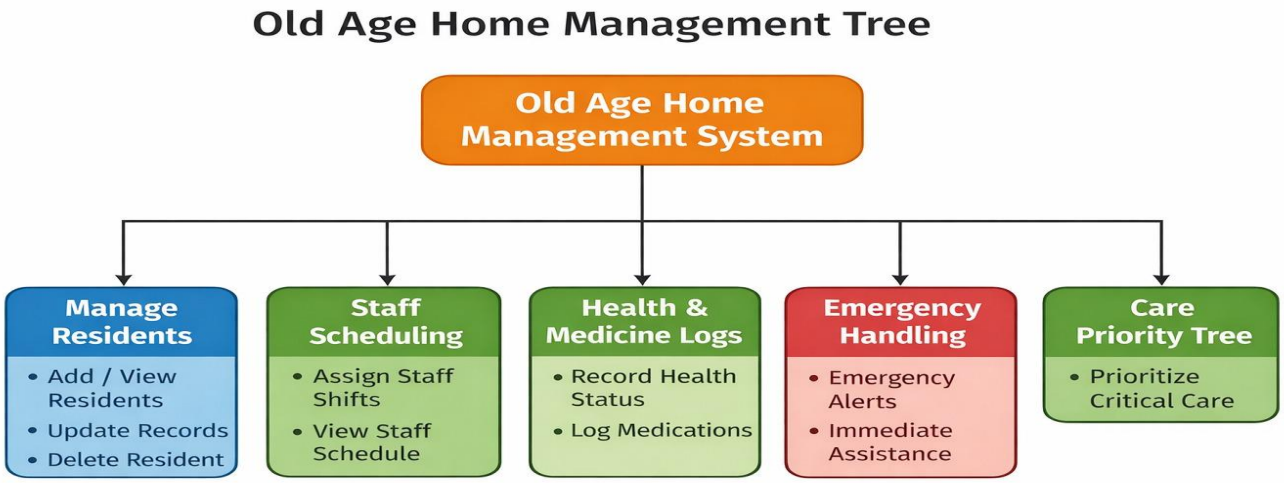
- Undo Medicine Update (Stack)
- Recursive Health Risk Evaluation
- BST-based Care Priority
- Circular Queue for Staff Rotation

6. Challenges Faced:

- Handling recursion effectively
- Managing dynamic health logs
- Console formatting for clear output

Future Enhancements:

- Emergency handling and Room assignment logic
- File saving and loading for persistence
- GUI version for better usability



Conclusion:

This project demonstrates modular design and the use of multiple data structures to solve real-world problems in elder care management. It highlights the importance of recursion, stacks, and trees in building efficient systems.

```cpp
#include <iostream>
using namespace std;
const int MAX = 50;
// ------Manage Residents Coding By Anmol Ihsan ----------
int ids[MAX];
int ages[MAX];
int health[MAX];
string names[MAX];
int rooms[MAX];
int countRes = 0;

void addResident() {
    cout << "\nEnter ID: ";
    cin >> ids[countRes];

    cout << "Enter Name: ";
    cin >> names[countRes];

    cout << "Enter Age: ";
    cin >> ages[countRes];

    cout << "Enter Room: ";
    cin >> rooms[countRes];

    cout << "Enter Health Score: ";
    cin >> health[countRes];

    countRes++;
    cout << "Resident added successfully!\n";
}

void searchByID() {
    int searchID;
    cout << "\nEnter ID to search: ";
    cin >> searchID;

    for (int i = 0; i < countRes; i++) {
        if (ids[i] == searchID) {
            cout << "\nResident Found:\n";
            cout << "ID: " << ids[i]
                 << " Name: " << names[i]
                 << " Age: " << ages[i]
                 << " Room: " << rooms[i]
                 << " Health: " << health[i] << endl;
            return;
        }
    }
    cout << "Resident not found!\n";
}

void sortByHealth() {
    for (int i = 0; i < countRes - 1; i++) {
        for (int j = 0; j < countRes - i - 1; j++) {
            if (health[j] > health[j + 1]) {
                swap(health[j], health[j + 1]);
                swap(ids[j], ids[j + 1]);
                swap(names[j], names[j + 1]);
                swap(ages[j], ages[j + 1]);
                swap(rooms[j], rooms[j + 1]);
            }
        }
    }
    cout << "Residents sorted by health score!\n";
}
void displayResidents() {
    if (countRes == 0) {
        cout << "No residents available.\n";
        return;
    }
    cout << "\nCurrent Residents:\n";
    cout << "-----------------------------------------------\n";
    cout << "ID\tName\tAge\tRoom\tHealth\n";
    cout << "-----------------------------------------------\n";

    for (int i = 0; i < countRes; i++) {
        cout << ids[i] << "\t"
             << names[i] << "\t"
             << ages[i] << "\t"
             << rooms[i] << "\t"
             << health[i] << endl;
    }
}
void manageResidents() {
    int choice;
    do {
        cout << "\n--- RESIDENT MANAGEMENT ---\n";
        cout << "1. Add New Resident\n";
        cout << "2. Search Resident by ID\n";
        cout << "3. Sort Residents by Health Score\n";
        cout << "4. Display ALL Residents\n";
        cout << "0. Back to Main Menu\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1: addResident(); break;
        case 2: searchByID(); break;
        case 3: sortByHealth(); break;
        case 4: displayResidents(); break;
        case 0: break;
        default: cout << "Invalid choice!\n";
        }
    } while (choice != 0);
}
// ------STAFF SCHEDULING Coding By Fatima Saheli----------

#define STAFF_SIZE 5

string staffQueue[STAFF_SIZE] = {
    "Nurse Anmol", "Nurse Fatima", "Nurse Mujtaba", "Nurse Ismail", "Dr. Ali"
};
int staffCount = 5;

void rotateDuty() {
    if (staffCount <= 1) {
        cout << "Not enough staff members to rotate!\n";
        return;
    }
    string firstStaff = staffQueue[0];

    for (int i = 0; i < staffCount - 1; i++) {
        staffQueue[i] = staffQueue[i + 1];
    }

    staffQueue[staffCount - 1] = firstStaff;

    cout << "Duty rotated successfully!\n";
}

void displayDutySchedule() {
    cout << "\n=== CURRENT STAFF DUTY SCHEDULE ===\n";
    cout << "Queue Order: FRONT -> ";
    for (int i = 0; i < staffCount; i++) {
        cout << staffQueue[i];
        if (i < staffCount - 1) {
            cout << " -> ";
        }
```

```
    }
    cout << " -> BACK\n\n";

    cout << "Detailed Schedule:\n";
    cout << "--------------------------------\n";
    for (int i = 0; i < staffCount; i++) {
        cout << "Position " << (i + 1) << ": " << staffQueue[i];
        if (i == 0) {
            cout << " [CURRENTLY ON DUTY]";
        }
        cout << endl;
    }
    cout << "--------------------------------\n";
}
void staffScheduler() {
    int choice;
    do {
        cout << "\n--- STAFF SCHEDULING ---\n";
        cout << "1. Rotate Duty\n";
        cout << "2. Display Duty Schedule\n";
        cout << "0. Back to Main Menu\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: rotateDuty(); break;
            case 2: displayDutySchedule(); break;
            case 0: break;
            default: cout << "Invalid choice!\n";
        }
    } while (choice != 0);
}
// ------Health & Medicine Logs Coding By Syed Mujtaba Hassan----------
struct HealthLog {
    int residentID;
    string date;
    string medicine;
    int dosage;
    string notes;
    int healthScore;
    HealthLog* next;

    HealthLog(int id, string d, string med, int dos, string n, int score) {
        residentID = id;
        date = d;
        medicine = med;
        dosage = dos;
        notes = n;
        healthScore = score;
        next = NULL;
    }
};
struct MedicineUpdate {
    int residentID;
    string oldMedicine;
    string newMedicine;
    int oldDosage;
    int newDosage;
    MedicineUpdate* next;

    MedicineUpdate(int id, string oldMed, string newMed, int oldDos, int ne
        residentID = id;
        oldMedicine = oldMed;
        newMedicine = newMed;
        oldDosage = oldDos;
        newDosage = newDos;
        next = NULL;
    }
};
```

```

HealthLog* healthLogsHead = NULL;
MedicineUpdate* undoStackTop = NULL;

// Linked List insertion at beginning.
void addHealthLog() {
    int id, dosage, score;
    string date, medicine, notes;

    cout << "\n--- ADD HEALTH LOG ---\n";
    cout << "Enter Resident ID: ";
    cin >> id;

    cout << "Enter Date (DD-MM-YYYY): ";
    cin >> date;

    cout << "Enter Medicine Name: ";
    cin.ignore();
    getline(cin, medicine);

    cout << "Enter Dosage (mg): ";
    cin >> dosage;

    cout << "Enter Notes: ";
    cin.ignore();
    getline(cin, notes);

    cout << "Enter Health Score (0-100): ";
    cin >> score;

    HealthLog* newLog = new HealthLog(id, date, medicine, dosage, notes, score);

    newLog->next = healthLogsHead;
    healthLogsHead = newLog;

    cout << "Health log added successfully!\n";
}
void displayHealthLogs() {
    if (healthLogsHead == NULL) {
        cout << "\nNo health logs available.\n";
        return;
    }
    cout << "\n--- ALL HEALTH LOGS ---\n";
    cout << "=========================================================================\n";
    cout << "ResID\tDate\t\tMedicine\t\tDosage\tScore\tNotes\n";
    cout << "=========================================================================\n";

    HealthLog* current = healthLogsHead;
    int count = 0;

    while (current != NULL) {
        cout << current->residentID << "\t"
             << current->date << "\t"
             << current->medicine;

        if (current->medicine.length() < 8) cout << "\t\t";
        else if (current->medicine.length() < 16) cout << "\t";

        cout << current->dosage << "mg\t"
             << current->healthScore << "\t"
             << current->notes << endl;

        current = current->next;
        count++;
    }
    cout << "=========================================================================\n";
    cout << "Total logs: " << count << endl;
}
```

```
void displayResidentLogs() {
    int searchID;
    cout << "\nEnter Resident ID to view logs: ";
    cin >> searchID;

    HealthLog* current = healthLogsHead;
    bool found = false;
    int count = 0;

    cout << "\n--- HEALTH LOGS FOR RESIDENT ID: " << searchID << "
    cout << "=================================================
    cout << "Date\t\tMedicine\tDosage\tScore\tNotes\n";
    cout << "=================================================

    while (current != NULL) {
        if (current->residentID == searchID) {
            found = true;
            count++;
            cout << current->date << "\t"
                 << current->medicine << "\t"
                 << current->dosage << "mg\t"
                 << current->healthScore << " \t"
                 << current->notes << endl;
        }
        current = current->next;
    }

    if (!found) {
        cout << "No health logs found for Resident ID: " << searchI
    } else {
        cout << "=================================================
        cout << "Total logs: " << count << endl;
    }
}
// Update medicine and push to undo stack
void updateMedicine() {
    int id, newDosage;
    string newMedicine;

    cout << "\n--- UPDATE MEDICINE ---\n";
    cout << "Enter Resident ID: ";
    cin >> id;

    HealthLog* current = healthLogsHead;
    string currentMedicine = "";
    int currentDosage = 0;

    while (current != NULL) {
        if (current->residentID == id) {
            currentMedicine = current->medicine;
            currentDosage = current->dosage;
            break;
        }
        current = current->next;
    }

    if (currentMedicine == "") {
        cout << "No health log found for Resident ID: " << id << en
        return;
    }

    cout << "Current Medicine: " << currentMedicine << " (" << curr
    cout << "Enter New Medicine Name: ";
    cin.ignore();
    getline(cin, newMedicine);

    cout << "Enter New Dosage (mg): ";
    cin >> newDosage;

    MedicineUpdate* update = new MedicineUpdate(id, currentMedicine, newMedicine, currentDosage, ne Dosage);
    update->next = undoStackTop;
    undoStackTop = update;

    current->medicine = newMedicine;
    current->dosage = newDosage;

    cout << "Medicine updated successfully! (Can be undone)\n";
}
void undoMedicineUpdate() {
    if (undoStackTop == NULL) {
        cout << "\nNo updates to undo!\n";
        return;
    }

    MedicineUpdate* update = undoStackTop;
    HealthLog* current = healthLogsHead;
    while (current != NULL) {
        if (current->residentID == update->residentID) {

            current->medicine = update->oldMedicine;
            current->dosage = update->oldDosage;

            cout << "\n--- UPDATE UNDONE ---\n";
            cout << "Resident ID: " << update->residentID << endl;
            cout << "Medicine reverted from '" << update->newMedicine
                 << "' to '" << update->oldMedicine << "'\n";
            cout << "Dosage reverted from " << update->newDosage
                 << "mg to " << update->oldDosage << "mg\n";

            undoStackTop = undoStackTop->next;
            delete update;
            return;
        }
        current = current->next;
    }

    cout << "Could not find corresponding health log!\n";
}

void displayUndoStack() {
    if (undoStackTop == NULL) {
        cout << "\nUndo stack is empty.\n";
        return;
    }

    cout << "\n--- UNDO STACK (Most Recent First) ---\n";
    cout << "=======================================\n";
    cout << "ResID\tOld Medicine\tNew Medicine\n";
    cout << "=======================================\n";

    MedicineUpdate* current = undoStackTop;
    int count = 0;

    while (current != NULL) {
        cout << current->residentID << "\t"
             << current->oldMedicine << " (" << current->oldDosage << "mg)\t"
             << current->newMedicine << " (" << current->newDosage << "mg)\n";
        current = current->next;
        count++;
    }

    cout << "=======================================\n";
    cout << "Updates available for undo: " << count << endl;
}
void evaluateHealthRiskRecursive(HealthLog* node) {
    if (node == NULL) {
        return;
    }
```

```
    evaluateHealthRiskRecursive(node->next);

    cout << "Resident ID " << node->residentID << " -> Score: " << node->healthSc

    if (node->healthScore < 50) {
        cout << "Status: Critical ";
    } else if (node->healthScore < 70) {
        cout << "Status: Moderate ";
    } else {
        cout << "Status: Stable ";
    }
    cout << endl;
}
void evaluateHealthRisk() {
    if (healthLogsHead == NULL) {
        cout << "\nNo health logs available for evaluation.\n";
        return;
    }
    cout << "\n--- EVALUATING HEALTH RISK (Recursively) ---\n";
    cout << "============================================\n";

    evaluateHealthRiskRecursive(healthLogsHead);

    cout << "============================================\n";
}
void healthMedicineLogs() {
    int choice;

    do {
        cout << "\n--- HEALTH & MEDICINE LOGS ---\n";
        cout << "1. Add Health Log (Linked List)\n";
        cout << "2. Display All Health Logs\n";
        cout << "3. Display Logs for Specific Resident\n";
        cout << "4. Update Medicine (Push to Stack)\n";
        cout << "5. Undo Last Medicine Update (Pop from Stack)\n";
        cout << "6. Display Undo Stack\n";
        cout << "7. Evaluate Health Risk (Recursion)\n";
        cout << "0. Back to Main Menu\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: addHealthLog(); break;
            case 2: displayHealthLogs(); break;
            case 3: displayResidentLogs(); break;
            case 4: updateMedicine(); break;
            case 5: undoMedicineUpdate(); break;
            case 6: displayUndoStack(); break;
            case 7: evaluateHealthRisk(); break;
            case 0: cout << "Returning to Main Menu...\n"; break;
            default: cout << "Invalid choice!\n";
        }
    } while (choice != 0);
}
// -------- CARE PRIORITY TREE (BST) by Ismail Khalid --------

struct TreeNode {
    int health;
    string name;
    TreeNode* left;
    TreeNode* right;
};

TreeNode* insertTree(TreeNode* root, int health, string name) {
    if (!root) {
        TreeNode* node = new TreeNode;
        node->health = health;
        node->name = name;
        node->left = node->right = NULL;
```

```
    int health;
    string name;
    TreeNode* left;
    TreeNode* right;
};

TreeNode* insertTree(TreeNode* root, int health, string name)
    if (!root) {
        TreeNode* node = new TreeNode;
        node->health = health;
        node->name = name;
        node->left = node->right = NULL;
        return node;
    }

    if (health < root->health)
        root->left = insertTree(root->left, health, name);
    else
        root->right = insertTree(root->right, health, name);

    return root;
}

void inorder(TreeNode* root) {
    if (!root) return;
    inorder(root->left);
    cout << root->name << " (Health: " << root->health << ")"<< endl;
    inorder(root->right);
}

void carePriorityTree() {
    TreeNode* root = NULL;

    for (int i = 0; i < countRes; i++) {
        root = insertTree(root, health[i], names[i]);
    }

    cout << "\n--- CARE PRIORITY (Low -> High) ---"<<endl;
    inorder(root);
}

int main() {
    int choice;

    do {
        cout << "\n==================================\n";
        cout << " OLD AGE HOME MANAGEMENT SYSTEM \n";
        cout << "==================================\n";
        cout << "1. Manage Residents\n";
        cout << "2. Staff Scheduling\n";
        cout << "3. Health & Medicine Logs\n";
        cout << "4. Care Priority Tree\n";
        cout << "0. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
           case 1: manageResidents(); break;
            case 2: staffScheduler(); break;
            case 3: healthMedicineLogs(); break;
            case 4: carePriorityTree(); break;
            case 0: cout << "\nExiting Program...\n"; break;
            default: cout << "\nInvalid choice!\n";
        }
    } while (choice != 0);

    return 0;
}
```

```
==================================
 OLD AGE HOME MANAGEMENT SYSTEM
==================================
1. Manage Residents
2. Staff Scheduling
3. Health & Medicine Logs
4. Care Priority Tree
0. Exit
Enter your choice: 1

--- RESIDENT MANAGEMENT ---
1. Add New Resident
2. Search Resident by ID
3. Sort Residents by Health Score
4. Display ALL Residents
0. Back to Main Menu
Enter your choice: 1

Enter ID: 111
Enter Name: Ali
Enter Age: 62
Enter Room: 20
Enter Health Score: 76
Resident added successfully!

--- RESIDENT MANAGEMENT ---
1. Add New Resident
2. Search Resident by ID
3. Sort Residents by Health Score
4. Display ALL Residents
0. Back to Main Menu
Enter your choice: 1

Enter ID: 222
Enter Name: Batool
Enter Age: 50
Enter Room: 10
Enter Health Score: 67
Resident added successfully!

--- RESIDENT MANAGEMENT ---
1. Add New Resident
2. Search Resident by ID
3. Sort Residents by Health Score
4. Display ALL Residents
0. Back to Main Menu
Enter your choice: 2

Enter ID to search: 222

Resident Found:
ID: 222 Name: Batool Age: 50 Room: 10 Health: 67
```

```
--- RESIDENT MANAGEMENT ---
1. Add New Resident
2. Search Resident by ID
3. Sort Residents by Health Score
4. Display ALL Residents
0. Back to Main Menu
Enter your choice: 3
Residents sorted by health score!

--- RESIDENT MANAGEMENT ---
1. Add New Resident
2. Search Resident by ID
3. Sort Residents by Health Score
4. Display ALL Residents
0. Back to Main Menu
Enter your choice: 4

Current Residents:
-----------------------------------------------
ID      Name    Age     Room    Health
-----------------------------------------------
222     Batool  50      10      67
111     Ali     62      20      76

--- RESIDENT MANAGEMENT ---
1. Add New Resident
2. Search Resident by ID
3. Sort Residents by Health Score
4. Display ALL Residents
0. Back to Main Menu
Enter your choice: 0

==================================
 OLD AGE HOME MANAGEMENT SYSTEM
==================================
1. Manage Residents
2. Staff Scheduling
3. Health & Medicine Logs
4. Care Priority Tree
0. Exit
Enter your choice: 2

--- STAFF SCHEDULING ---
1. Rotate Duty
2. Display Duty Schedule
0. Back to Main Menu
Enter your choice: 1
Duty rotated successfully!
```

```
--- STAFF SCHEDULING ---
1. Rotate Duty
2. Display Duty Schedule
0. Back to Main Menu
Enter your choice: 2

=== CURRENT STAFF DUTY SCHEDULE ===
Queue Order: FRONT -> Nurse Fatima -> Nurse Mujtaba -> Nurse Ismail -> Dr. Ali -> Nurse Anmol -> BACK

Detailed Schedule:
--------------------------------
Position 1: Nurse Fatima [CURRENTLY ON DUTY]
Position 2: Nurse Mujtaba
Position 3: Nurse Ismail
Position 4: Dr. Ali
Position 5: Nurse Anmol
--------------------------------

--- STAFF SCHEDULING ---
1. Rotate Duty
2. Display Duty Schedule
0. Back to Main Menu
Enter your choice: 0

==================================
 OLD AGE HOME MANAGEMENT SYSTEM
==================================
1. Manage Residents
2. Staff Scheduling
3. Health & Medicine Logs
4. Care Priority Tree
0. Exit
Enter your choice: 3

--- HEALTH & MEDICINE LOGS ---
1. Add Health Log (Linked List)
2. Display All Health Logs
3. Display Logs for Specific Resident
4. Update Medicine (Push to Stack)
5. Undo Last Medicine Update (Pop from Stack)
6. Display Undo Stack
7. Evaluate Health Risk (Recursion)
0. Back to Main Menu
Enter your choice: 1

--- ADD HEALTH LOG ---
Enter Resident ID: 111
Enter Date (DD-MM-YYYY): 12-12-1990
Enter Medicine Name: Metformin
Enter Dosage (mg): 140
Enter Notes: type 2 diabetes
Enter Health Score (0-100): 76
Health log added successfully!
```

```
--- HEALTH & MEDICINE LOGS ---
1. Add Health Log (Linked List)
2. Display All Health Logs
3. Display Logs for Specific Resident
4. Update Medicine (Push to Stack)
5. Undo Last Medicine Update (Pop from Stack)
6. Display Undo Stack
7. Evaluate Health Risk (Recursion)
0. Back to Main Menu
Enter your choice: 1

--- ADD HEALTH LOG ---
Enter Resident ID: 222
Enter Date (DD-MM-YYYY): 2-4-1998
Enter Medicine Name: Amlodipine
Enter Dosage (mg): 120
Enter Notes: high blood pressure
Enter Health Score (0-100): 67
Health log added successfully!

--- HEALTH & MEDICINE LOGS ---
1. Add Health Log (Linked List)
2. Display All Health Logs
3. Display Logs for Specific Resident
4. Update Medicine (Push to Stack)
5. Undo Last Medicine Update (Pop from Stack)
6. Display Undo Stack
7. Evaluate Health Risk (Recursion)
0. Back to Main Menu
Enter your choice: 2

--- ALL HEALTH LOGS ---
=================================================================================
ResID   Date            Medicine                Dosage  Score   Notes
=================================================================================
222     2-4-1998        Amlodipine      120mg   67      high blood pressure
111     12-12-1990      Metaformin      140mg   76      type 2 diabetes
=================================================================================
Total logs: 2

--- HEALTH & MEDICINE LOGS ---
1. Add Health Log (Linked List)
2. Display All Health Logs
3. Display Logs for Specific Resident
4. Update Medicine (Push to Stack)
5. Undo Last Medicine Update (Pop from Stack)
6. Display Undo Stack
7. Evaluate Health Risk (Recursion)
0. Back to Main Menu
Enter your choice: 3
```

```
Enter Resident ID to view logs: 222

--- HEALTH LOGS FOR RESIDENT ID: 222 ---
============================================================
Date            Medicine        Dosage  Score   Notes
============================================================
2-4-1998        Amlodipine      120mg   67      high blood pressure
============================================================
Total logs: 1

--- HEALTH & MEDICINE LOGS ---
1. Add Health Log (Linked List)
2. Display All Health Logs
3. Display Logs for Specific Resident
4. Update Medicine (Push to Stack)
5. Undo Last Medicine Update (Pop from Stack)
6. Display Undo Stack
7. Evaluate Health Risk (Recursion)
0. Back to Main Menu
Enter your choice: 4

--- UPDATE MEDICINE ---
Enter Resident ID: 111
Current Medicine: Metaformin (140mg)
Enter New Medicine Name: TZDs
Enter New Dosage (mg): 120
Medicine updated successfully! (Can be undone)

--- HEALTH & MEDICINE LOGS ---
1. Add Health Log (Linked List)
2. Display All Health Logs
3. Display Logs for Specific Resident
4. Update Medicine (Push to Stack)
5. Undo Last Medicine Update (Pop from Stack)
6. Display Undo Stack
7. Evaluate Health Risk (Recursion)
0. Back to Main Menu
Enter your choice: 5

--- UPDATE UNDONE ---
Resident ID: 111
Medicine reverted from 'TZDs' to 'Metaformin'
Dosage reverted from 120mg to 140mg

--- HEALTH & MEDICINE LOGS ---
1. Add Health Log (Linked List)
2. Display All Health Logs
3. Display Logs for Specific Resident
4. Update Medicine (Push to Stack)
5. Undo Last Medicine Update (Pop from Stack)
6. Display Undo Stack
7. Evaluate Health Risk (Recursion)
0. Back to Main Menu
Enter your choice: 6
```

```
Enter your choice: 6

Undo stack is empty.

--- HEALTH & MEDICINE LOGS ---
1. Add Health Log (Linked List)
2. Display All Health Logs
3. Display Logs for Specific Resident
4. Update Medicine (Push to Stack)
5. Undo Last Medicine Update (Pop from Stack)
6. Display Undo Stack
7. Evaluate Health Risk (Recursion)
0. Back to Main Menu
Enter your choice: 7

--- EVALUATING HEALTH RISK (Recursively) ---
============================================
Resident ID 111 -> Score: 76 ->Status: Stable
Resident ID 222 -> Score: 67 ->Status: Moderate
============================================

--- HEALTH & MEDICINE LOGS ---
1. Add Health Log (Linked List)
2. Display All Health Logs
3. Display Logs for Specific Resident
4. Update Medicine (Push to Stack)
5. Undo Last Medicine Update (Pop from Stack)
6. Display Undo Stack
7. Evaluate Health Risk (Recursion)
0. Back to Main Menu
Enter your choice: 0
Returning to Main Menu...

====================================
 OLD AGE HOME MANAGEMENT SYSTEM
====================================
1. Manage Residents
2. Staff Scheduling
3. Health & Medicine Logs
4. Care Priority Tree
0. Exit
Enter your choice: 0

Exiting Program...

--------------------------------
Process exited after 400.4 seconds with return value 0
Press any key to continue . . .
```