# MLM Simulator

Data Structures Project Report

Course: CPE 112: Programming with Data Structures

Semester: 2/2024

Instructors: Asst. Prof. Dr. Natasha Dejdumrong, Dr. Aye Hninn Khine, and Aj. Mr. Naveed

Submission Date: 5 March 2025

Group Members:

| Name | Student ID | Role |
|---|---|---|
| Peeranat Ngamkiatkajorn | 67070503429 | Team Leader |
| | | System Architect |
| Paemika Yongwittayakul | 67070503469 | Quality Assurance |
| Khongpop Manowanna | 67070503488 | C Developer |

## 1. Introduction

This project implements a multi-level marketing structure using Tree, Doubly Linked List and Hash Map. The system manages agents in multi-level marketing structures by storing each agent's ID as a key. Each node in table links to the tree structure which enables quick insertion, deletion and searching operations. Users are allowed to make a payment; the program calculates commission and compensation to an agent and up to 3 level of the agent's upline. The goal is to provide an efficient and scalable solution for managing agents in a multi-level marketing structure.

## 2. Project Requirements

2.1 Functional Requirements:

- Add new agent to the structure

- Remove and agent

- Update agent's commission rate

- Query specific agent details

- Visualize the MLM structure

- Simulate a client payment with commission flow

2.2 Non-Functional Requirements:

- Easy to use interface

- Fast look up and updates using Hash Map

- Saving multi-level structure data in a csv file and load up when start

## 3. Data Structures Overview

The primary data structure used is Tree. Each node contains:

- Agent ID (int)

- Upline ID (int)

- Pointer to upline node

- Pointer to downline nodes

- Downlines count (int)

- Personal sales (float)

- Commission earnings (float)

- Self commission rate (float)

This structure allows flexible hierarchical relationships among agents

The secondary data structure used is Hash Map. Each node contains:

- Agent ID (int)

- Pointer to corresponding tree node

- Pointer to next node in case of collision

Hash map was chosen because it allows for access to the tree node pointer to further do efficient search, insert, and delete operations in O(1) time for Hash Map

## 4. Algorithm Design

Main Functions:

- create_agent(agent ID, commission rate): Adds a new agent under a specific upline.

- remove_agent(agent ID): Remove an agent from both the tree and hash map.

- update_agent(agent ID, commission rate): Update an agent’s commission rate.

- query_agent(agent ID): Search for an agent's data.

- agent_tree(agent hash map pointer): Displays the entire MLM structure using preorder traversal.

- client_payment(agent ID, pay amount): Distributes commissions to the agent and up to 3 levels of uplines based on respective commission rates.

## 5. Complexity Analysis

| Operation | Data Structure | Time Complexity |
|---|---|---|
| Insertion | Hash Map + Tree | O(1) + O(1) |
| Deletion | Hash Map + Tree | O(1) + O(n) |
| Search (by ID) | Hash Map | O(1) |
| MLM Tree Traversal | Tree | O(n) |
| Commission Calculation | Tree | O(3) |

## 6. Implementation Details

Programming Language: C

Input: Command-line interface with prompt-based option

Output: Text-based display of MLM structure and agent data

Testing: Conduct a test on key functions like agent creation, deletion and commission calculation.

Memory management: Use dynamic memory allocation (malloc/free) and ensured proper memory free after removed agents or structures

## 7. Challenges & Lessons Learned

Challenges:

- Implementing a deletion in tree with multiple child nodes while maintaining hierarchical integrity
- Linking the hash map to the tree effectively without memory leaks
- Managing memory during complex operations like deletion or tree traversal

Lessons Learned:

- Understanding pointer manipulation and dynamic memory in C
- Importance of modular and clean code for large projects

- Debugging recursive structures and avoiding segmentation faults

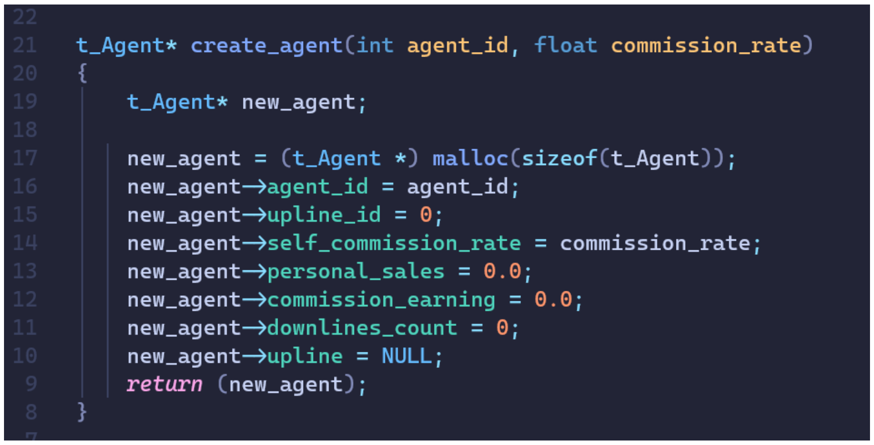
## REFERENCES

- Instructor lectures on Tree, Linked List, and Hash Map
- YouTube tutorial on Tree, Linked List, and Hash Map
- ChatGPT

## Appendices

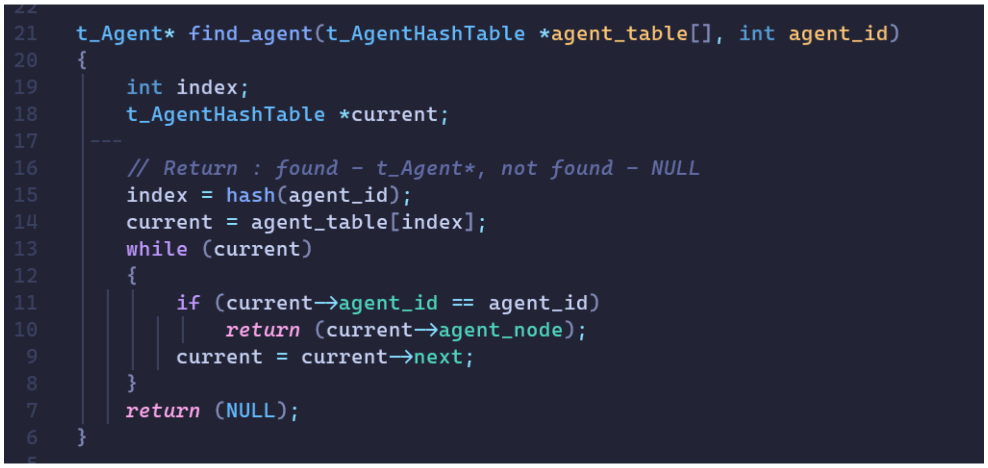
Appendix A: Sample Code Snippet

1. Create Agent



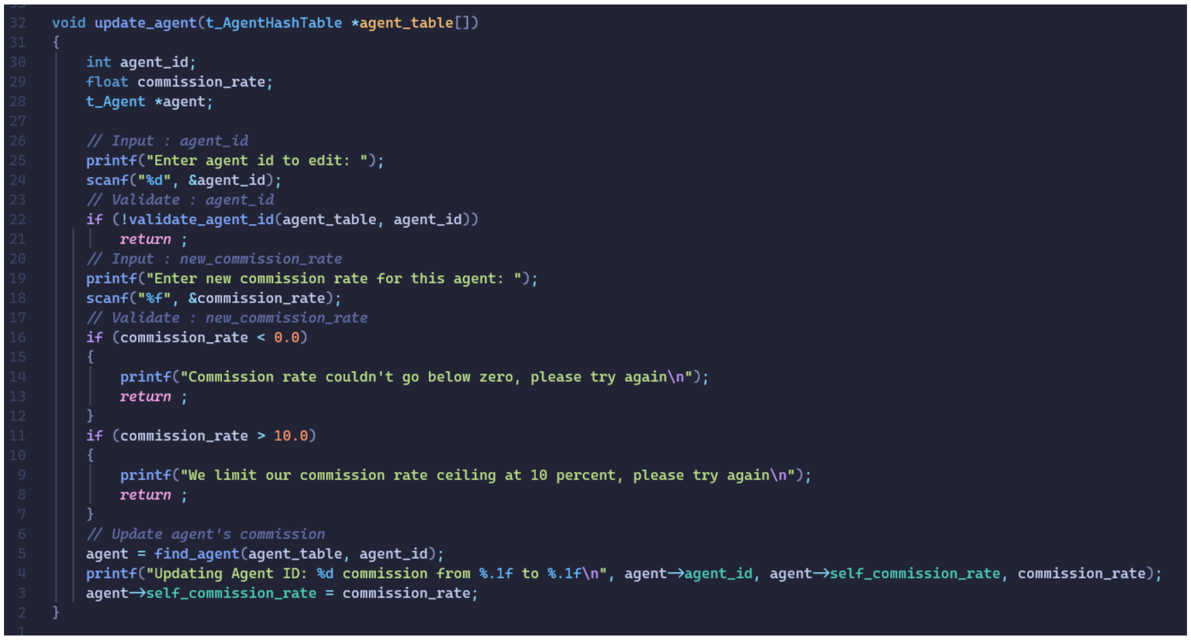
```c
t_Agent* create_agent(int agent_id, float commission_rate)
{
    t_Agent* new_agent;

    new_agent = (t_Agent *) malloc(sizeof(t_Agent));
    new_agent->agent_id = agent_id;
    new_agent->upline_id = 0;
    new_agent->self_commission_rate = commission_rate;
    new_agent->personal_sales = 0.0;
    new_agent->commission_earning = 0.0;
    new_agent->downlines_count = 0;
    new_agent->upline = NULL;
    return (new_agent);
}
```

2. Find an Agent



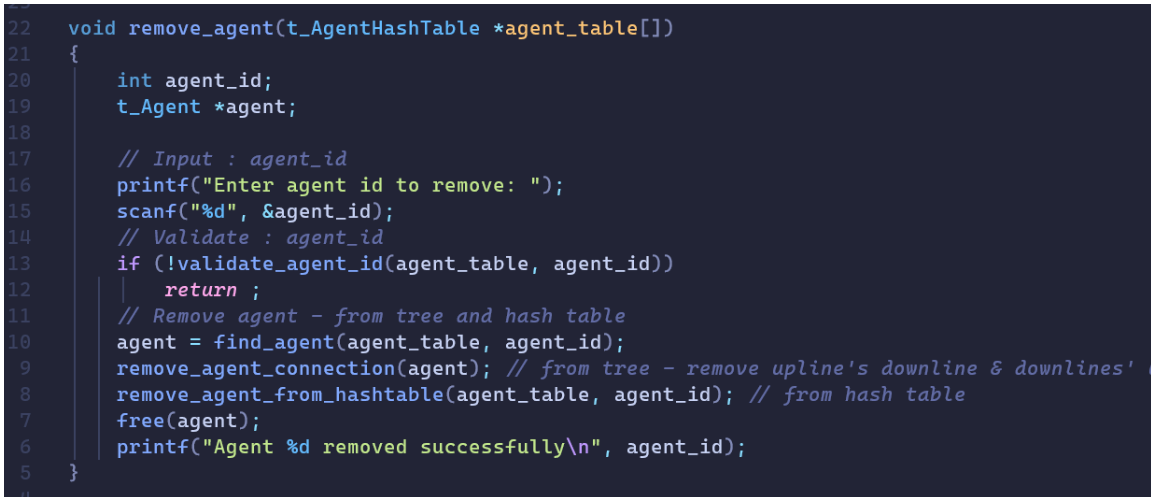
```c
t_Agent* find_agent(t_AgentHashTable *agent_table[], int agent_id)
{
    int index;
    t_AgentHashTable *current;

    // Return : found - t_Agent*, not found - NULL
    index = hash(agent_id);
    current = agent_table[index];
    while (current)
    {
        if (current->agent_id == agent_id)
            return (current->agent_node);
        current = current->next;
    }
    return (NULL);
}
```

3. Update Agent's Commission



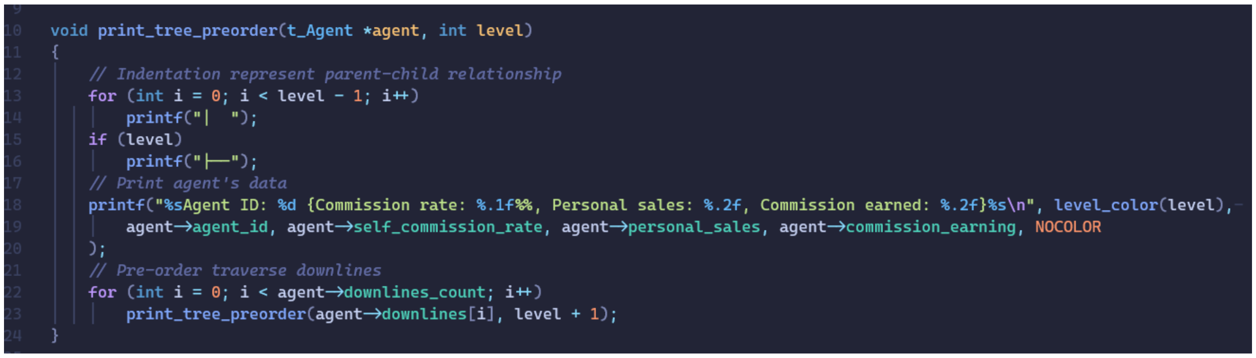
```c
void update_agent(t_AgentHashTable *agent_table[])
{
    int agent_id;
    float commission_rate;
    t_Agent *agent;

    // Input : agent_id
    printf("Enter agent id to edit: ");
    scanf("%d", &agent_id);
    // Validate : agent_id
    if (!validate_agent_id(agent_table, agent_id))
        return ;
    // Input : new_commission_rate
    printf("Enter new commission rate for this agent: ");
    scanf("%f", &commission_rate);
    // Validate : new_commission_rate
    if (commission_rate < 0.0)
    {
        printf("Commission rate couldn't go below zero, please try again\n");
        return ;
    }
    if (commission_rate > 10.0)
    {
        printf("We limit our commission rate ceiling at 10 percent, please try again\n");
        return ;
    }
    // Update agent's commission
    agent = find_agent(agent_table, agent_id);
    printf("Updating Agent ID: %d commission from %.1f to %.1f\n", agent->agent_id, agent->self_commission_rate, commission_rate);
    agent->self_commission_rate = commission_rate;
}
```

4. Remove Agent



```c
void remove_agent(t_AgentHashTable *agent_table[])
{
    int agent_id;
    t_Agent *agent;

    // Input : agent_id
    printf("Enter agent id to remove: ");
    scanf("%d", &agent_id);
    // Validate : agent_id
    if (!validate_agent_id(agent_table, agent_id))
        return ;
    // Remove agent - from tree and hash table
    agent = find_agent(agent_table, agent_id);
    remove_agent_connection(agent); // from tree - remove upline's downline & downlines'
    remove_agent_from_hashtable(agent_table, agent_id); // from hash table
    free(agent);
    printf("Agent %d removed successfully\n", agent_id);
}
```

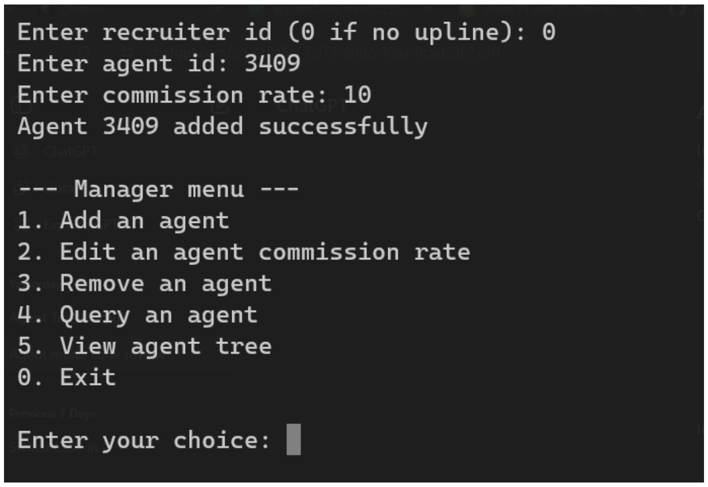
5. Display structure



```c
void print_tree_preorder(t_Agent *agent, int level)
{
    // Indentation represent parent-child relationship
    for (int i = 0; i < level - 1; i++)
        printf("|  ");
    if (level)
        printf("├─");
    // Print agent's data
    printf("%sAgent ID: %d {Commission rate: %.1f%%, Personal sales: %.2f, Commission earned: %.2f}%s\n", level_color(level),
        agent->agent_id, agent->self_commission_rate, agent->personal_sales, agent->commission_earning, NOCOLOR
    );
    // Pre-order traverse downlines
    for (int i = 0; i < agent->downlines_count; i++)
        print_tree_preorder(agent->downlines[i], level + 1);
}
```

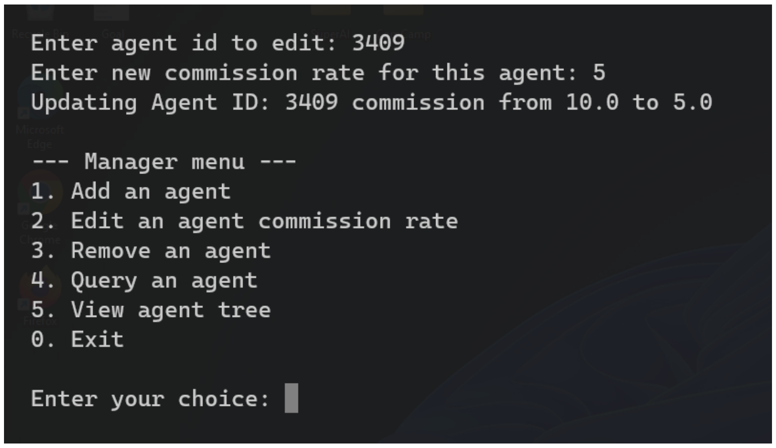
Appendix B: Sample Input/Output

1. Add agent to the structure



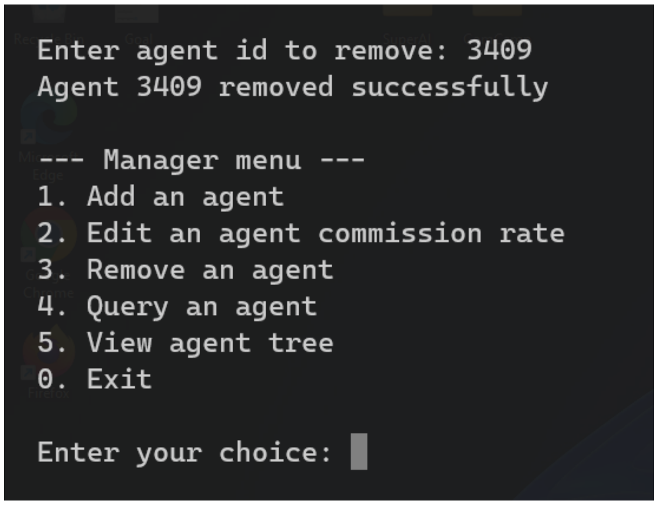
```
Enter recruiter id (0 if no upline): 0
Enter agent id: 3409
Enter commission rate: 10
Agent 3409 added successfully

--- Manager menu ---
1. Add an agent
2. Edit an agent commission rate
3. Remove an agent
4. Query an agent
5. View agent tree
0. Exit

Enter your choice:
```

2. Update agent's commission rate



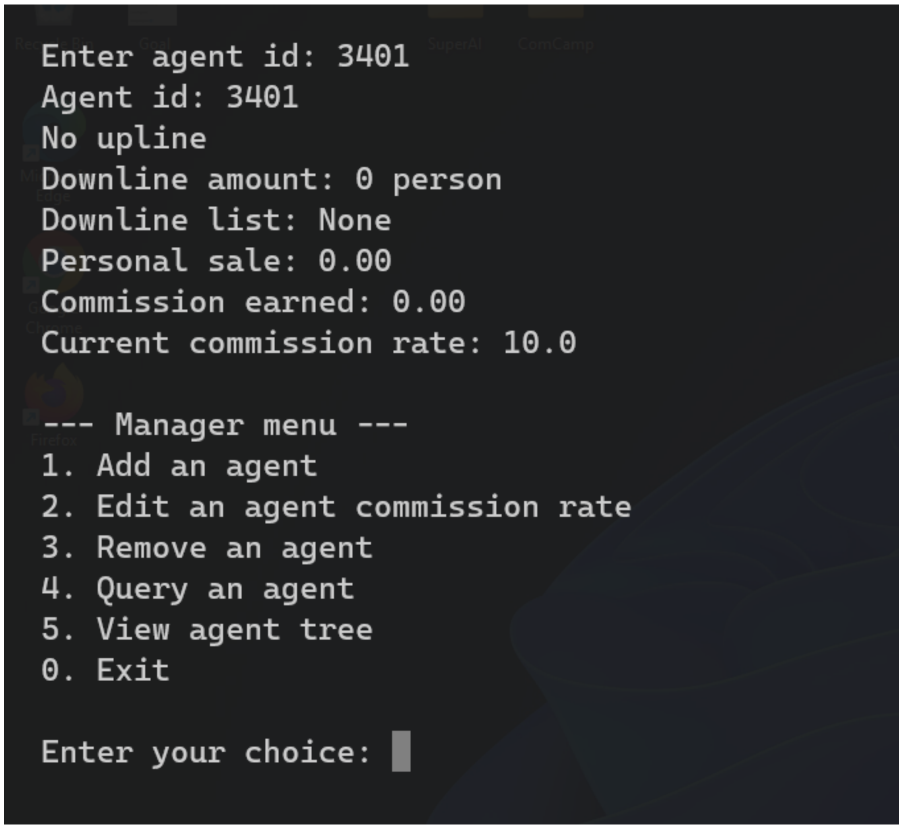
```
Enter agent id to edit: 3409
Enter new commission rate for this agent: 5
Updating Agent ID: 3409 commission from 10.0 to 5.0

--- Manager menu ---
1. Add an agent
2. Edit an agent commission rate
3. Remove an agent
4. Query an agent
5. View agent tree
0. Exit

Enter your choice:
```

3. Remove an agent from the structure



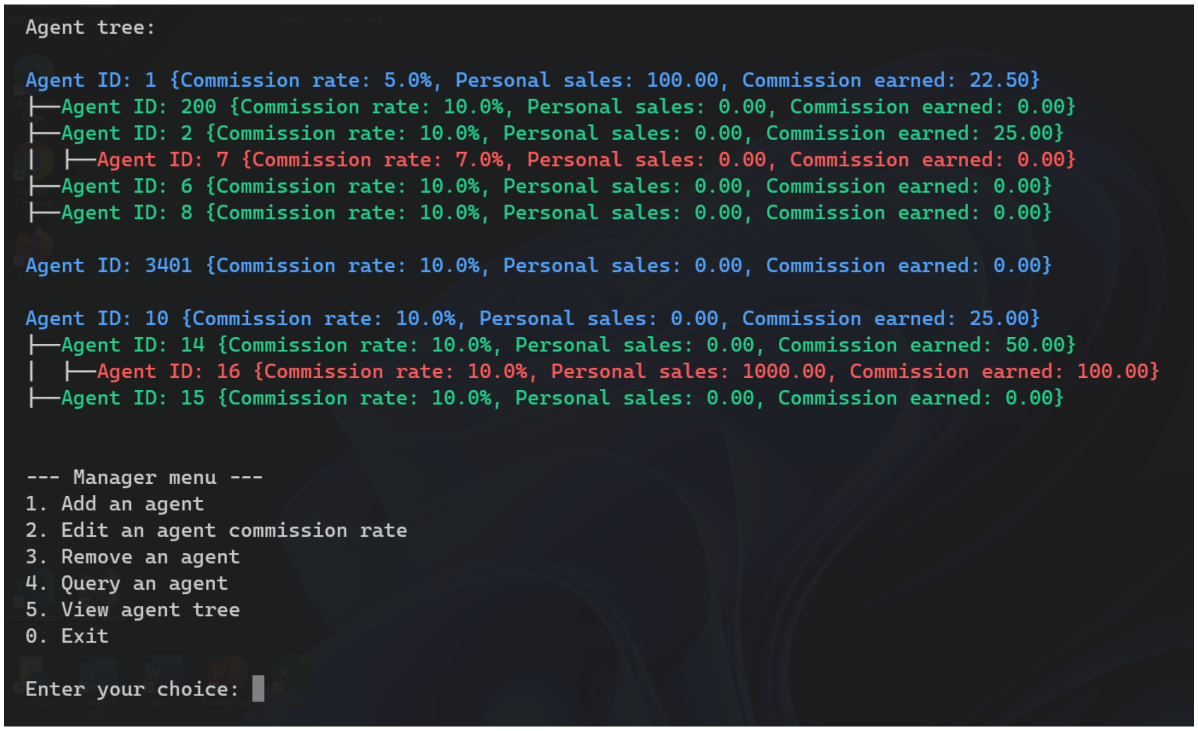
```
Enter agent id to remove: 3409
Agent 3409 removed successfully

--- Manager menu ---
1. Add an agent
2. Edit an agent commission rate
3. Remove an agent
4. Query an agent
5. View agent tree
0. Exit

Enter your choice:
```

4. Query an agent



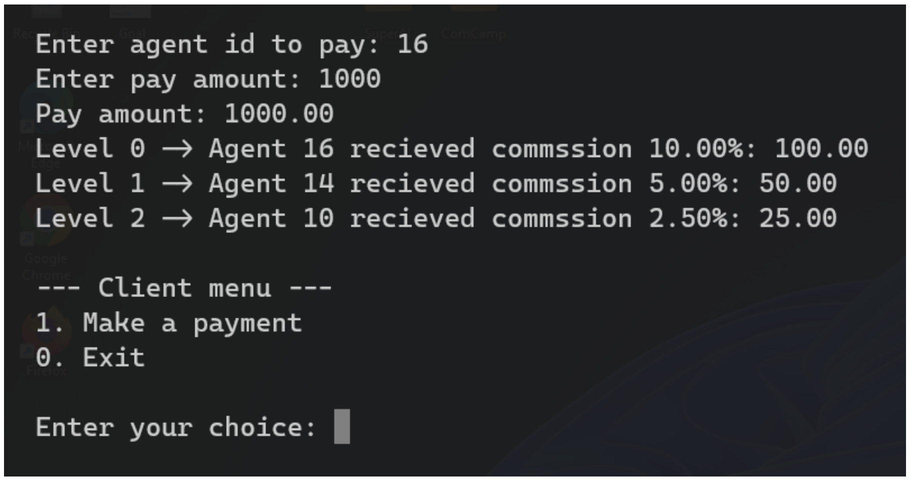
```
Enter agent id: 3401
Agent id: 3401
No upline
Downline amount: 0 person
Downline list: None
Personal sale: 0.00
Commission earned: 0.00
Current commission rate: 10.0

--- Manager menu ---
1. Add an agent
2. Edit an agent commission rate
3. Remove an agent
4. Query an agent
5. View agent tree
0. Exit

Enter your choice: 
```

5. Display structure



```
Agent tree:

Agent ID: 1 {Commission rate: 5.0%, Personal sales: 100.00, Commission earned: 22.50}
├─Agent ID: 200 {Commission rate: 10.0%, Personal sales: 0.00, Commission earned: 0.00}
├─Agent ID: 2 {Commission rate: 10.0%, Personal sales: 0.00, Commission earned: 25.00}
|  ├─Agent ID: 7 {Commission rate: 7.0%, Personal sales: 0.00, Commission earned: 0.00}
├─Agent ID: 6 {Commission rate: 10.0%, Personal sales: 0.00, Commission earned: 0.00}
├─Agent ID: 8 {Commission rate: 10.0%, Personal sales: 0.00, Commission earned: 0.00}

Agent ID: 3401 {Commission rate: 10.0%, Personal sales: 0.00, Commission earned: 0.00}

Agent ID: 10 {Commission rate: 10.0%, Personal sales: 0.00, Commission earned: 25.00}
├─Agent ID: 14 {Commission rate: 10.0%, Personal sales: 0.00, Commission earned: 50.00}
|  ├─Agent ID: 16 {Commission rate: 10.0%, Personal sales: 1000.00, Commission earned: 100.00}
├─Agent ID: 15 {Commission rate: 10.0%, Personal sales: 0.00, Commission earned: 0.00}


--- Manager menu ---
1. Add an agent
2. Edit an agent commission rate
3. Remove an agent
4. Query an agent
5. View agent tree
0. Exit

Enter your choice: 
```

6. Make a payment


```
Enter agent id to pay: 16
Enter pay amount: 1000
Pay amount: 1000.00
Level 0 --> Agent 16 recieved commssion 10.00%: 100.00
Level 1 --> Agent 14 recieved commssion 5.00%: 50.00
Level 2 --> Agent 10 recieved commssion 2.50%: 25.00

--- Client menu ---
1. Make a payment
0. Exit

Enter your choice: 
```