### Project Documentation

### I.Project Title:

**Jumanji Network Management System**

### II. Project Description

* **Overview:** This project involves developing a C program to manage a multi-level marketing network structured as a binary tree. The program facilitates the addition of agents, displays the network structure, and shows downlines. It helps manage agent relationships and track commissions efficiently.
* **Objective:** The objective of the system is to manage agents and their commissions in a hierarchical network, making it easier to track downlines and commissions for multi-level marketing organizations. The goal is to create a user-friendly interface for managing agents in an efficient manner.
* **Scope:**

The project is a C program designed to manage agent networks using a binary tree structure. It focuses on basic functionalities, such as adding agents, displaying the network structure, and showing downlines. However, it is limited to basic error handling and lacks persistent data storage, advanced commission features, and memory management. This project is intended for small-scale use or educational purposes and is not suitable for large-scale implementations or integrations with external systems.

### III. Features

**1. Add Agents (Downlines):** Users can add agents to the network, placing them in the binary tree. The program increases the commission for the parent agent each time a left and right downline are added.

**2. Display Network Structure:** The system visualizes the entire network structure, displaying each agent and their commission in a tree-like format.

**3. Search and View Agent Downlines:** Allows searching for an agent by name, and displays their direct downlines and commission information.

**4. Commission Calculation:** The system calculates and displays the total commission earned by all members in the network.

**5. Interactive Downline Addition:** Enables users to add left and right downlines interactively with custom names.

**6. User Authentication:** The program includes a basic authentication system, ensuring that only authorized users can access the network management features.

### IV. Technologies Used

* **Programming Languages:** C
* **Tools:** GCC compiler, VS Code/Codeblocks for development, GitHub for version control
* **Libraries:** None (standalone C program)
* **Database:** None (in-memory storage)

### V. Project Structure

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define COMMISSION\_PER\_PAIR 500

typedef struct Member {

char name[50];

int commission;

int isFirstInvite;

struct Member\* left;

struct Member\* right;

} Member;

void clearScreen();

Member\* createMember(const char\* name);

void displayNetwork(Member\* root, int level);

Member\* searchMember(Member\* root, const char\* name);

void displayDownlines(Member\* member);

void addDownlineInteractive(Member\* root);

void displayTotalCommission(Member\* root);

void freeNetwork(Member\* root);

void clearScreen() {

system("clear");

}

Member\* createMember(const char\* name) {

Member\* newMember = (Member\*)malloc(sizeof(Member));

if (newMember == NULL) {

printf("Memory allocation failed.\n");

exit(1);

}

strcpy(newMember->name, name);

newMember->commission = 0;

newMember->isFirstInvite = 1;

newMember->left = newMember->right = NULL;

return newMember;

}

void displayNetwork(Member\* root, int level) {

if (root == NULL) return;

for (int i = 0; i < level; i++) printf(" ");

printf("|| %s (Commission: P%d) ||\n", root->name, root->commission);

displayNetwork(root->left, level + 1);

displayNetwork(root->right, level + 1);

}

Member\* searchMember(Member\* root, const char\* name) {

if (root == NULL) return NULL;

if (strcmp(root->name, name) == 0) return root;

Member\* found = searchMember(root->left, name);

if (found != NULL) return found;

return searchMember(root->right, name);

}

void displayDownlines(Member\* member) {

if (member == NULL) return;

printf("\nMember: %s\n", member->name);

printf("Commission: P%d\n", member->commission);

if (member->left) {

printf("Left Downline: %s\n", member->left->name);

} else {

printf("Left Downline: None\n");

}

if (member->right) {

printf("Right Downline: %s\n", member->right->name);

} else {

printf("Right Downline: None\n");

}

}

void addDownline(Member\* parent, char\* leftName, char\* rightName, Member\* root) {

if (leftName != NULL && strlen(leftName) > 0) {

parent->left = createMember(leftName);

}

if (rightName != NULL && strlen(rightName) > 0) {

parent->right = createMember(rightName);

}

if (parent->left != NULL && parent->right != NULL) {

if (parent == root && root->isFirstInvite) {

root->commission += COMMISSION\_PER\_PAIR;

root->isFirstInvite = 0;

} else {

parent->commission += COMMISSION\_PER\_PAIR;

root->commission += COMMISSION\_PER\_PAIR;

}

}

}

void addDownlineInteractive(Member\* root) {

char parentName[50], leftName[50], rightName[50];

clearScreen();

printf("Enter the name of the parent member: ");

scanf("%49s", parentName);

while (getchar() != '\n');

Member\* parent = searchMember(root, parentName);

if (parent == NULL) {

printf("Parent member not found.\n");

return;

}

printf("Enter the name of the left downline (or press Enter to skip): ");

fgets(leftName, 50, stdin);

leftName[strcspn(leftName, "\n")] = 0;

printf("Enter the name of the right downline (or press Enter to skip): ");

fgets(rightName, 50, stdin);

rightName[strcspn(rightName, "\n")] = 0;

addDownline(parent, leftName, rightName, root);

}

int calculateTotalCommission(Member\* root) {

if (root == NULL) return 0;

return root->commission + calculateTotalCommission(root->left) + calculateTotalCommission(root->right);

}

void displayTotalCommission(Member\* root) {

int totalCommission = calculateTotalCommission(root);

printf("\nTotal Commission Earned by All Members: P%d\n", totalCommission);

}

void freeNetwork(Member\* root) {

if (root == NULL) return;

freeNetwork(root->left);

freeNetwork(root->right);

free(root);

}

int authenticateUser() {

char username[50], password[50];

const char validUsername[] = "admin";

const char validPassword[] = "password";

clearScreen();

printf("\n\t\t========================================\n");

printf("\t\t\| JUMANJI NETWORK MANAGEMENT SYSTEM |\n");

printf("\t\t| \033[1;34m--- Login Required ---\033[0m |\n");

printf("\t\t========================================\n");

printf("\033[1;36m\t\tEnter Username: \033[0m");

scanf("%49s", username);

printf("\033[1;36m\t\tEnter Password: \033[0m");

scanf("%49s", password);

if (strcmp(username, validUsername) == 0 && strcmp(password, validPassword) == 0) {

printf("\n\t\t\033[1;32mLogin successful!\033[0m\n");

return 1;

} else {

printf("\n\t\t\033[1;31mInvalid username or password. Exiting...\033[0m\n");

return 0;

}

}

int main() {

if (!authenticateUser()) {

return 0;

}

char rootName[50];

clearScreen();

printf("\033[1;32mEnter the name of the root member: \033[0m");

scanf("%49s", rootName);

Member\* root = createMember(rootName);

int choice;

while (1) {

clearScreen();

printf("\n\t\t+=======================================+\n");

printf("\t\t| \033[1;34m--- JUMANJI Network Management ---\033[0m |\n");

printf("\t\t+=======================================+\n");

printf("\t\t| 1. Display Network Tree |\n");

printf("\t\t| 2. Search Member and Display Downlines|\n");

printf("\t\t| 3. Add Downlines |\n");

printf("\t\t| 4. Display Total Commission |\n");

printf("\t\t| 5. Exit |\n");

printf("\t\t+=======================================+\n");

printf("\t\tEnter your choice: ");

scanf("%d", &choice);

clearScreen();

switch (choice) {

case 1:

printf("\nNetwork Tree:\n");

displayNetwork(root, 0);

break;

case 2: {

char searchName[50];

printf("Enter the name of the member: ");

scanf("%49s", searchName);

Member\* member = searchMember(root, searchName);

if (member) {

displayDownlines(member);

} else {

printf("Member not found.\n");

}

break;

}

case 3:

addDownlineInteractive(root);

break;

case 4:

displayTotalCommission(root);

break;

case 5:

printf("Exiting...\n");

freeNetwork(root);

exit(0);

default:

printf("Invalid choice. Please try again.\n");

}

printf("\nPress Enter to continue...");

while (getchar() != '\n');

getchar();

}

return 0;

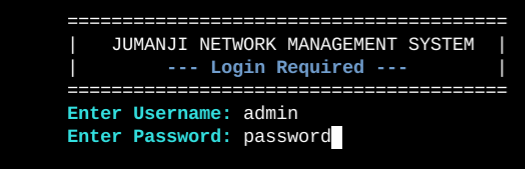
}

### 

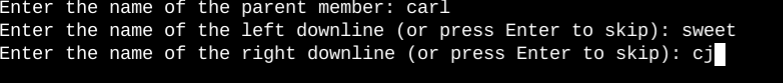
### 

### VI. Usage

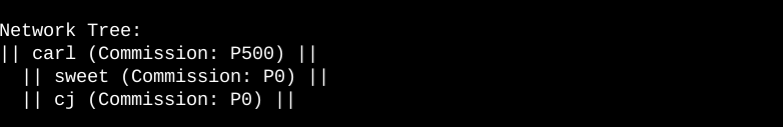
**1. Log-in Page**



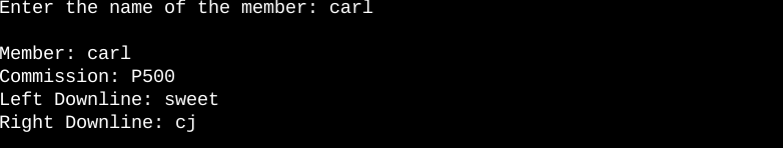
**2. Add Downlines**

****

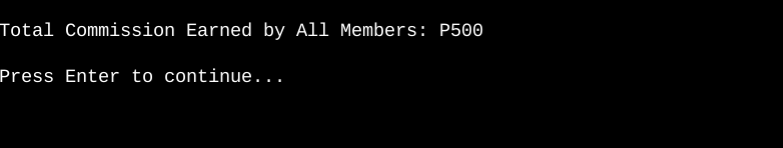
**3. Display Network Tree**

****

**4. Search and Display Downlines**

****

**5. Display Total Commisions**

****

### VII. Testing

1. **Adding Agents:** Test the ability to add agents and check if commissions are correctly updated.
2. **Downline Management:** Verify that downlines are correctly displayed for each agent, including edge cases (e.g., no downlines).
3. **Commission Calculation:** Ensure that the commission is calculated accurately for each agent based on successful downline additions.
4. **Network Structure:** Test the display of the network hierarchy to ensure it represents the agent relationships correctly.

### VIII. Contribution

Steps for contributing to the project:

* Fork the repository.
* Create a new branch: git checkout -b feature-branch.
* Commit changes: git commit -m "Add feature".
* Push to GitHub and submit a pull request.

### IX. Acknowledgments

* Special thanks to the C programming community and educational resources that helped with implementing binary trees and agent network management.

### 

### 

### 

### 

### X. Contact Information

* **Authors:** Carl James O. Patenio, CJ Pesole, Kevin Duone, Michelle Ivy, Sweet Rose Balansag
* **Email:** carl46436@gmail.com
* **GitHub:** https://github.com/Carl46436/Jumanji-Network-Management-.git