**Project Documentation Template**

**Project Title** Jumanji Network Management System

**Project Description**

* **Overview**: The Jumanji Network Management System is a C-based hierarchical member management tool that helps users create, display, and manage a network of members with downlines. Each member earns a commission for successfully adding both a left and right downline.
* **Objective**: The project aims to streamline the process of managing multi-level marketing networks by automating member addition, commission calculation, and network visualization.
* **Scope**:  
  + The system focuses on a binary tree structure, where each member can have up to two downlines (left and right).
  + Commission is awarded only when both downlines are added.
  + The project assumes a maximum of one level depth for adding new downlines interactively.

**Features**

1. **Create Member**: Initialize a new member with a name and zero commission.
2. **Add Downlines**: Add left and right downlines to a specified parent member.
3. **Commission Calculation**: Automatically calculate and update the commission of a member upon successful downline addition.
4. **Display Network Tree**: Visualize the entire network hierarchy in a tree-like format.
5. **Search and Display Downlines**: Search for a specific member and display their immediate downlines and commission.
6. **Interactive Downline Addition**: Allow users to interactively add downlines to existing members.
7. **Menu-driven Interface**: Provides a simple, intuitive command-line menu for managing the network.

**Technologies Used**

* **Programming Language**: C
* **Tools**: GCC compiler, VS Code, Git

**Project Structure**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define COMMISSION\_PER\_PAIR 500

typedef struct Member {

char name[50];

int commission;

struct Member\* left;

struct Member\* right;

} Member;

Member\* createMember(char\* name) {

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

strcpy(newMember->name, name);

newMember->commission = 0;

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

return newMember;

}

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

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) {

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

}

}

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 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);

}

void addDownlineInteractive(Member\* root) {

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

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

scanf("%s", parentName);

// Search for the parent node

Member\* current = root;

if (strcmp(current->name, parentName) != 0) {

if (current->left && strcmp(current->left->name, parentName) == 0) {

current = current->left;

} else if (current->right && strcmp(current->right->name, parentName) == 0) {

current = current->right;

} else {

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

return;

}

}

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

scanf("%s", leftName);

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

scanf("%s", rightName);

addDownline(current, leftName, rightName);

}

int main() {

Member\* root = createMember("Carl");

addDownline(root, "CJ", "Sweet");

addDownline(root->left, "Ivy", "John");

addDownline(root->right, "Kevin", "Michael");

int choice;

while (1) {

printf("\n--- Jumanji Network Management ---\n");

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

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

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

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

system("clear");

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

displayNetwork(root, 0);

break;

case 2:

system("clear");

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

char searchName[50];

scanf("%s", searchName);

if (strcmp(root->name, searchName) == 0) {

displayDownlines(root);

} else if (root->left && strcmp(root->left->name, searchName) == 0) {

displayDownlines(root->left);

} else if (root->right && strcmp(root->right->name, searchName) == 0) {

displayDownlines(root->right);

} else {

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

}

break;

case 3:

system("clear");

addDownlineInteractive(root);

break;

case 4:

system("clear");

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

exit(0);

break;

default:

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

}

}

return 0;

}

**Usage**

* + **Compilation**: Use the following command to compile the project:  
      
     gcc main.c -o jumanji\_network
  + **Execution**: Run the compiled program using:  
      
     ./jumanji\_network

1. **Menu Options**:  
   * **1. Display Network Tree**: Displays the current hierarchical structure of the network.
   * **2. Search Member and Display Downlines**: Prompts the user to enter a member's name and displays their downlines.
   * **3. Add Downlines**: Allows the user to add downlines to an existing member.
   * **4. Exit**: Terminates the program.
   * **Example Output**:  
       
      **Sample Network Tree**:  
       
      || Carl (Commission: P1000)||
   * || CJ (Commission: P500)||
   * || Ivy (Commission: P0)||
   * || John (Commission: P0)||
   * || Sweet (Commission: P500)||
   * || Kevin (Commission: P0)||
   * || Michael (Commission: P0)||

**Testing**

1. **Manual Testing**:  
   * Verify that adding both left and right downlines increases the member’s commission by 500.
   * Check the correct display of the network tree after each operation.
   * Test search functionality with both existing and non-existing member names.
2. **Automated Testing**: Not implemented.

**Contributions**

1. Fork the repository.
   * Clone the repository to your local machine.  
      git clone https://github.com/username/jumanji\_network.git
   * Create a new branch for your feature.  
      git checkout -b feature-branch
   * Commit your changes.  
      git commit -m "Add feature"
   * Push the changes to your forked repository.  
      git push origin feature-branch
2. Submit a pull request.

**Acknowledgments**

* Thanks to the C programming community for inspiration.
* Special thanks to online forums and tutorials for guidance on binary tree structures.

**Contact Information**

* **Author**: 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-System