Department of Computer Engineering has student's club named 'Pinnacle Club'. Students of second, third and Similarly one may cancel the membership of club. First node is reserved for president of club and last node is reserved for secretary of club. Write C++ program to maintain club member‘s information using singly linked list. Store student PRN and Name. Write functions to: 1. Add and delete the members as well as president or even secretary. 2. Compute total number of members of club 3. Display members 4. Two linked lists exist for two divisions. Concatenate two lists.  
  
#include <iostream>

#include <string>

using namespace std;

// Node structure

struct Node {

int PRN;

string name;

Node\* next;

Node(int p, string n) {

PRN = p;

name = n;

next = nullptr;

}

};

// Singly Linked List Class

class PinnacleClub {

private:

Node\* head;

public:

PinnacleClub() {

head = nullptr;

}

// Add member (President, Secretary, or general member)

void addMember(int prn, string name, string position) {

Node\* newNode = new Node(prn, name);

if (position == "President") {

newNode->next = head;

head = newNode;

} else if (position == "Secretary") {

if (head == nullptr) {

head = newNode;

} else {

Node\* temp = head;

while (temp->next != nullptr)

temp = temp->next;

temp->next = newNode;

}

} else {

if (head == nullptr) {

cout << "Add President first!" << endl;

delete newNode;

return;

}

Node\* temp = head;

while (temp->next != nullptr && temp->next->next != nullptr)

temp = temp->next;

newNode->next = temp->next;

temp->next = newNode;

}

}

// Delete member by PRN

void deleteMember(int prn) {

if (head == nullptr) {

cout << "List is empty!" << endl;

return;

}

if (head->PRN == prn) {

Node\* temp = head;

head = head->next;

delete temp;

cout << "Member deleted successfully." << endl;

return;

}

Node\* temp = head;

while (temp->next != nullptr && temp->next->PRN != prn)

temp = temp->next;

if (temp->next == nullptr) {

cout << "Member not found!" << endl;

} else {

Node\* toDelete = temp->next;

temp->next = temp->next->next;

delete toDelete;

cout << "Member deleted successfully." << endl;

}

}

// Count members

int countMembers() {

int count = 0;

Node\* temp = head;

while (temp != nullptr) {

count++;

temp = temp->next;

}

return count;

}

// Display members

void displayMembers() {

if (head == nullptr) {

cout << "Club has no members!" << endl;

return;

}

Node\* temp = head;

while (temp != nullptr) {

cout << "PRN: " << temp->PRN << ", Name: " << temp->name << endl;

temp = temp->next;

}

}

// Concatenate two linked lists

void concatenate(PinnacleClub& other) {

if (head == nullptr) {

head = other.head;

} else {

Node\* temp = head;

while (temp->next != nullptr)

temp = temp->next;

temp->next = other.head;

}

other.head = nullptr;

}

};

// Main function

int main() {

PinnacleClub divA, divB;

int choice;

do {

cout << "\n--- Pinnacle Club Menu ---\n";

cout << "1. Add Member\n";

cout << "2. Delete Member\n";

cout << "3. Display Members\n";

cout << "4. Count Members\n";

cout << "5. Concatenate Two Divisions\n";

cout << "6. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1: {

int prn;

string name, position;

cout << "Enter PRN: ";

cin >> prn;

cin.ignore();

cout << "Enter Name: ";

getline(cin, name);

cout << "Enter Position (President/Secretary/Member): ";

cin >> position;

divA.addMember(prn, name, position);

break;

}

case 2: {

int prn;

cout << "Enter PRN to delete: ";

cin >> prn;

divA.deleteMember(prn);

break;

}

case 3:

cout << "\nMembers of Division A:\n";

divA.displayMembers();

break;

case 4:

cout << "\nTotal Members in Division A: " << divA.countMembers() << endl;

break;

case 5: {

cout << "Enter members for Division B: \n";

int n, prn;

string name, position;

cout << "How many members to add in Division B? ";

cin >> n;

for (int i = 0; i < n; i++) {

cout << "\nEnter PRN: ";

cin >> prn;

cin.ignore();

cout << "Enter Name: ";

getline(cin, name);

cout << "Enter Position (President/Secretary/Member): ";

cin >> position;

divB.addMember(prn, name, position);

}

divA.concatenate(divB);

cout << "Divisions concatenated successfully.\n";

break;

}

case 6:

cout << "Exiting Program...\n";

break;

default:

cout << "Invalid choice! Please try again.\n";

}

} while (choice != 6);

return 0;

}