```
#include <iostream>
#include <string>

using namespace std;

struct Node {
    int prn;
    string name;
    Node* next;

    Node(int p, string n) : prn(p), name(n), next(nullptr) {}
};

class PinnacleClub {
    Node* head;

public:
    PinnacleClub() : head(nullptr) {}

    void addPresident(int prn, string name) {
        Node* new_node = new Node(prn, name);
        new_node->next = head;
        head = new_node;
    }

    void addSecretary(int prn, string name) {
        Node* new_node = new Node(prn, name);
        if (!head) {
            head = new_node;
        } else {
            Node* temp = head;
            while (temp->next) {
```

```
            temp = temp->next;
        }
        temp->next = new_node;
    }
}

void addMember(int prn, string name) {
    Node* new_node = new Node(prn, name);
    if (!head) {
        cout << "Add president first.\n";
        return;
    }
    Node* temp = head;
    while (temp->next && temp->next->next) {
        temp = temp->next;
    }
    new_node->next = temp->next;
    temp->next = new_node;
}

void deletePresident() {
    if (!head) {
        cout << "No members to delete.\n";
        return;
    }
    Node* temp = head;
    head = head->next;
    delete temp;
}

void deleteSecretary() {
    if (!head || !head->next) {
        cout << "No members to delete.\n";
        return;
    }
    Node* temp = head;
    while (temp->next && temp->next->next) {
        temp = temp->next;
    }
    Node* to_delete = temp->next;
    temp->next = nullptr;
    delete to_delete;
}

void deleteMember(int prn) {
    if (!head) {
        cout << "No members to delete.\n";
        return;
    }
    if (head->prn == prn) {
        deletePresident();
        return;
    }
    Node* temp = head;
    while (temp->next && temp->next->prn != prn) {
        temp = temp->next;
    }
    if (temp->next) {
        Node* to_delete = temp->next;
        temp->next = temp->next->next;
        delete to_delete;
    } else {
        cout << "Member not found.\n";
    }
```

```
}

int countMembers() {
    int count = 0;
    Node* temp = head;
    while (temp) {
        count++;
        temp = temp->next;
    }
    return count;
}

void displayMembers() {
    Node* temp = head;
    while (temp) {
        cout << "PRN: " << temp->prn << ", Name: " << temp->name << endl;
        temp = temp->next;
    }
}

void concatenate(PinnacleClub& other) {
    if (!head) {
        head = other.head;
    } else {
        Node* temp = head;
        while (temp->next) {
            temp = temp->next;
        }
        temp->next = other.head;
    }
    other.head = nullptr;
}
```

```
    ~PinnacleClub() {
        while (head) {
            Node* temp = head;
            head = head->next;
            delete temp;
        }
    }
};

int main() {
    PinnacleClub club1, club2;

    club1.addPresident(1, "President1");
    club1.addMember(2, "Member1");
    club1.addSecretary(3, "Secretary1");

    club2.addPresident(4, "President2");
    club2.addMember(5, "Member2");
    club2.addSecretary(6, "Secretary2");

    cout << "Club 1 members:\n";
    club1.displayMembers();

    cout << "\nClub 2 members:\n";
    club2.displayMembers();

    club1.concatenate(club2);

    cout << "\nAfter concatenation:\n";
    club1.displayMembers();
    cout << "\nTotal members in Club 1: " << club1.countMembers() << endl;


    club1.deletePresident();
    cout << "\nAfter deleting President:\n";
    club1.displayMembers();


    club1.deleteSecretary();
    cout << "\nAfter deleting Secretary:\n";
    club1.displayMembers();


    club1.deleteMember(2);
    cout << "\nAfter deleting Member with PRN 2:\n";
    club1.displayMembers();


    return 0;
}
```