```
#include <iostream>
#include <string>
using namespace std;

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

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

class PinnacleClub {
private:
    Node* head;
    Node* tail;

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

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

    void addSecretary(int prn, string name) {
        Node* newSecretary = new Node(prn, name);
        if (!head) {
            head = tail = newSecretary;
        } else {
            tail->next = newSecretary;
            tail = newSecretary;
        }
    }

    void addMember(int prn, string name) {
        if (!head) {
            cout << "Add a president first.\n";
            return;
        }
        Node* newMember = new Node(prn, name);
        if (tail) {
```

```
        tail->next = newMember;
        tail = newMember;
    }
}

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

void deleteSecretary() {
    if (!head) {
        cout << "No members in the club.\n";
        return;
    }
    if (head == tail) {
        delete head;
        head = tail = nullptr;
        return;
    }
    Node* current = head;
    while (current->next != tail) {
        current = current->next;
    }
    delete tail;
    tail = current;
    tail->next = nullptr;
}

void deleteMember(int prn) {
    if (!head) {
        cout << "No members to delete.\n";
        return;
    }
    if (head->prn == prn) {
        deletePresident();
        return;
    }
```

```
    if (tail->prn == prn) {
        deleteSecretary();
        return;
    }
    Node* current = head;
    Node* previous = nullptr;
    while (current && current->prn != prn) {
        previous = current;
        current = current->next;
    }
    if (current) {
        previous->next = current->next;
        delete current;
    } else {
        cout << "Member with PRN " << prn << " not found.\n";
    }
}

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

void displayMembers() const {
    if (!head) {
        cout << "No members in the club.\n";
        return;
    }
    Node* current = head;
    cout << "Club Members:\n";
    cout << "President: ";
    if (current) {
        cout << current->prn << " - " << current->name << "\n";
        current = current->next;
    }
    while (current && current != tail) {
        cout << "Member: " << current->prn << " - " << current->name << "\n";
        current = current->next;
    }
```

```
        cout << "Secretary: ";
        if (tail) {
            cout << tail->prn << " - " << tail->name << "\n";
        }
    }

    void concatenate(PinnacleClub& otherClub) {
        if (!head) {
            head = otherClub.head;
            tail = otherClub.tail;
        } else if (otherClub.head) {
            tail->next = otherClub.head;
            tail = otherClub.tail;
        }
        otherClub.head = nullptr;
        otherClub.tail = nullptr;
    }

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

int main() {
    PinnacleClub divisionA;
    divisionA.addPresident(1, "Alice");
    divisionA.addMember(2, "Bob");
    divisionA.addMember(3, "Charlie");
    divisionA.addSecretary(4, "Daisy");

    cout << "\nDivision A Members:\n";
    divisionA.displayMembers();
    cout << "Total Members: " << divisionA.countMembers() << endl;

    divisionA.deleteMember(2); // Delete a member
    cout << "\nAfter deleting a member:\n";
    divisionA.displayMembers();
    cout << "Total Members: " << divisionA.countMembers() << endl;

    PinnacleClub divisionB;
```

```
    divisionB.addPresident(5, "Eve");
    divisionB.addMember(6, "Frank");
    divisionB.addSecretary(7, "Grace");

    cout << "\nDivision B Members:\n";
    divisionB.displayMembers();
    cout << "Total Members: " << divisionB.countMembers() << endl;

    divisionA.concatenate(divisionB);
    cout << "\nAfter concatenating Division B into Division A:\n";
    divisionA.displayMembers();
    cout << "Total Members in Division A after concatenation: " << divisionA.countMembers() <<
endl;

    return 0;
}
```