/*

Department of Computer Engineering has student's club named 'Pinnacle Club'. Students of Second, third and final year of department can be granted membership on request. 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

a) Add and delete the members as well as president or even secretary.

b) Compute total number of members of club

c) Display members

d) Display list in reverse order using recursion

e) Two linked lists exists for two divisions. Concatenate two lists

*/

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

class list;
```

```
class node {
   int prn;
   string name;
   node *next;

public:
   node(int x, string nm) {
      prn = x;
      next = NULL;
      name = nm;
   }

   friend class list;
};

class list {
   node *start;

public:
   list() {
      start = NULL;
   }

   void create();
   void display();
```

```
    void insertAtBeginning();
    void insertAtEnd();
    void insertAfter();
    void deleteAtFirst();
    void deleteByValue();
    void deleteAtEnd();
    int computeTotal();
    void sortList();
    void concatList(list &q1);
    void displayRev(node *t);
    bool reverseDisplay();
};

void list::displayRev(node *t) {
    if (t == NULL)
        return;
    else {
        displayRev(t->next);
        cout << "\nPRN NO:" << t->prn << " Name: " << t->name;
    }
}

void list::create() {
    int no;
    string nam;
```

```
    if (start == NULL) {
        cout << "Enter PRN number: ";
        cin >> no;
        cout << "Enter name: ";
        cin >> nam;
        cout << nam;
        start = new node(no, nam);
        cout << "\n=============== List Created ===============";
    } else {
        cout << "\nList is already created.";
    }
}

void list::display() {
    node *t;
    t = start;
    if (start == NULL)
        cout << "\nList is Empty";
    else {
        cout << "\n=============== List: ===============\n";
        while (t != NULL) {
            cout << t->prn << " " << t->name << " \n";
            t = t->next;
        }
    }
```

```
}

void list::insertAtBeginning() {
   int no;
   string nam;
   node *temp;
   if (start == NULL) {
      create();
   } else {
      cout << "\nEnter PRN Number: ";
      cin >> no;
      cout << "Enter Name: ";
      cin >> nam;
      temp = new node(no, nam);
      temp->next = start;
      start = temp;
      cout << "Inserted " << temp->name << " at the beginning.";
   }
}

void list::insertAtEnd() {
   int no;
   string nam;
   node *t;
   if (start == NULL)
```

```
        create();
    else {
        cout << "\nEnter PRN Number: ";
        cin >> no;
        cout << "Enter Name: ";
        cin >> nam;
        t = start;
        while (t->next != NULL)
            t = t->next;
        node *p = new node(no, nam);
        t->next = p;
    }
}

void list::insertAfter() {
    int prev_no;
    cout << "\nEnter PRN No. after which you want to insert: ";
    cin >> prev_no;
    node *t;
    t = start;
    string nam;
    int flag = 0, no;
    while (t != NULL) {
        if (t->prn == prev_no) {
            flag = 1;
```

```
            break;
        }
        t = t->next;
    }
    if (flag == 1) {
        node *p;
        cout << "\nEnter PRN Number: ";
        cin >> no;
        cout << "Enter Name: ";
        cin >> nam;
        p = new node(no, nam);
        p->next = t->next;
        t->next = p;
    } else {
        cout << "\n" << prev_no << " is not in the list.";
    }
}

void list::deleteAtFirst() {
    node *t;
    if (start == NULL)
        cout << "\nClub is Empty..";
    else {
        t = start;
        start = start->next;
```

```
        t->next = NULL;
        delete t;
        cout << "\nPresident deleted..";
    }
}

void list::deleteByValue() {
    int no, flag = 0;
    node *t, *prev;
    if (start == NULL)
        cout << "\nList/Club is empty";
    else {
        cout << "\nEnter PRN No. of member to be deleted: ";
        cin >> no;
        t = start->next;
        while (t->next != NULL) {
            if (t->prn == no) {
                flag = 1;
                break;
            }
            prev = t;
            t = t->next;
        }
        if (flag == 1) {
            prev->next = t->next;
```

```
            t->next = NULL;
            delete t;
            cout << "\nMember with PRN No: " << no << " is deleted.";
        } else
            cout << "\nMember not found in List./President or Secretary cannot be deleted.";
    }
}

void list::deleteAtEnd() {
    node *t, *prev;
    t = start;
    if (start == NULL)
        cout << "\nClub is Empty..";
    else {
        while (t->next != NULL) {
            prev = t;
            t = t->next;
        }
        prev->next = NULL;
        delete t;
        cout << "\nSecretary Deleted.";
    }
}

int list::computeTotal() {
```

```
    node *t;
    int count = 0;
    t = start;
    if (start == NULL) {
        cout << "\nList is empty.";
        return 0;
    }
    while (t != NULL) {
        count++;
        t = t->next;
    }
    return count;
}

void list::sortList() {
    node *i, *j, *last = NULL;
    int tprn;
    string tname;
    if (start == NULL) {
        cout << "\nList is empty.";
        return;
    }
    for (i = start; i->next != NULL; i = i->next) {
        for (j = start; j->next != last; j = j->next) {
            if ((j->prn) > (j->next->prn)) {
```

```
            tprn = j->prn;
            tname = j->name;
            j->prn = j->next->prn;
            j->name = j->next->name;
            j->next->prn = tprn;
            j->next->name = tname;
          }
        }
    }
    cout << "\nList is sorted.";
    display();
}

void list::concatList(list &q1) {
    node *t, *p;
    t = q1.start;
    if (t == NULL) {
        cout << "\nList 2 is empty";
        return;
    }
    p = start;
    while (p->next != NULL) {
        p = p->next;
    }
    p->next = t;
```

```
    q1.start = NULL;
    cout << "\nAfter concatenation list:\n";
    display();
}

bool list::reverseDisplay() {
    if (start == NULL)
        return false;
    node *temp = start;
    displayRev(temp);
    return true;
}

int main() {
    list *l;
    int choice, selectList;
    list l1, l2;
    l = &l1;

X:
    cout << "\nSelect List\n1.List 1\n2.List 2\nEnter choice: ";
    cin >> selectList;
    if (selectList == 1) {
        l = &l1;
    } else if (selectList == 2) {
```

```
    l = &l2;
} else {
    cout << "\nWrong list Number.";
    goto X;
}

do {
    cout << "\n1. Create\n2. Insert President\n3. Insert secretary\n4. Insert after position(member)\n";
    cout << "5. Display list\n6. Delete President\n7. Delete Secretary\n8. Delete Member\n";
    cout << "9. Find total No. of members\n10. Sort list\n11. Reselect List\n";
    cout << "12. Combine lists\n13. Reverse Display\n0. Exit\nEnter your choice: ";
    cin >> choice;

    switch (choice) {
        case 1: l->create();
                break;
        case 2: l->insertAtBeginning();
                break;
        case 3: l->insertAtEnd();
                break;
        case 4: l->insertAfter();
                break;
        case 5: l->display();
                break;
        case 6: l->deleteAtFirst();
```

```
            break;
        case 7: l->deleteAtEnd();
            break;
        case 8: l->deleteByValue();
            break;
        case 9: cout << "\nTotal members (including President & Secretary): " << l->computeTotal();
            break;
        case 10: l->sortList();
            break;
        case 11: goto X;
            break;
        case 12: l1.concatList(l2);
            break;
        case 13: l->reverseDisplay();
            break;
        default: cout << "Wrong choice";
      }
   } while (choice != 0);

   cout << "\n================ GOOD BYE ================\n";
   return 0;
}
```