Practical no.7

**Program code**:

#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), name(nm), next(NULL) {}

friend class list; // Grant list access to private members

};

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() {

if (start == NULL) return false;

displayRev(start);

return true;

}

};

void list::displayRev(node \*t) {

if (t == NULL) return;

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;

start = new node(no, nam);

cout << "\n=============== List Created ===============";

} else {

cout << "\nList is already created.";

}

}

void list::display() {

node \*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;

cout << "\nEnter PRN Number: ";

cin >> no;

cout << "Enter Name: ";

cin >> nam;

node \*temp = new node(no, nam);

temp->next = start;

start = temp;

cout << "Inserted " << temp->name << " at the beginning.";

}

void list::insertAtEnd() {

int no;

string nam;

cout << "\nEnter PRN Number: ";

cin >> no;

cout << "Enter Name: ";

cin >> nam;

node \*p = new node(no, nam);

if (start == NULL) {

start = p;

} else {

node \*t = start;

while (t->next != NULL) {

t = t->next;

}

t->next = p;

}

}

void list::insertAfter() {

int prev\_no;

cout << "\nEnter PRN No. after which you want to insert: ";

cin >> prev\_no;

node \*t = start;

string nam;

int no;

bool found = false;

while (t != NULL) {

if (t->prn == prev\_no) {

found = true;

break;

}

t = t->next;

}

if (found) {

cout << "\nEnter PRN Number: ";

cin >> no;

cout << "Enter Name: ";

cin >> nam;

node \*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() {

if (start == NULL) {

cout << "\nClub is Empty..";

} else {

node \*t = start;

start = start->next;

delete t;

cout << "\nPresident deleted..";

}

}

void list::deleteByValue() {

int no;

cout << "\nEnter PRN No. of member to be deleted: ";

cin >> no;

if (start == NULL) {

cout << "\nList/Club is empty";

return;

}

node \*t = start;

node \*prev = NULL;

if (t->prn == no) {

start = t->next;

delete t;

cout << "\nMember with PRN No: " << no << " is deleted.";

return;

}

while (t != NULL && t->prn != no) {

prev = t;

t = t->next;

}

if (t != NULL) {

prev->next = t->next;

delete t;

cout << "\nMember with PRN No: " << no << " is deleted.";

} else {

cout << "\nMember not found in List.";

}

}

void list::deleteAtEnd() {

if (start == NULL) {

cout << "\nClub is Empty..";

} else {

node \*t = start;

node \*prev = NULL;

while (t->next != NULL) {

prev = t;

t = t->next;

}

if (prev == NULL) {

start = NULL; // Only one node

} else {

prev->next = NULL;

}

delete t;

cout << "\nSecretary Deleted.";

}

}int list::computeTotal() {

node \*t = start;

int count = 0;

while (t != NULL) {

count++;

t = t->next;

}

return count;

}

void list::sortList() {

if (start == NULL) {

cout << "\nList is empty.";

return;

}

for (node \*i = start; i != NULL; i = i->next) {

for (node \*j = start; j->next != NULL; j = j->next) {

if (j->prn > j->next->prn) {

swap(j->prn, j->next->prn);

swap(j->name, j->next->name);

}

}

}

cout << "\nList is sorted.";

display();

}

void list::concatList(list &q1) {

if (q1.start == NULL) {

cout << "\nList 2 is empty";

return;

}

if (start == NULL) {

start = q1.start;

} else {

node \*p = start;

while (p->next != NULL) {

p = p->next;

}

p->next = q1.start;

}

q1.start = NULL; // Set second list to null

cout << "\nAfter concatenation list: \n";

display();

}

int main() {

list l1, l2;

list \*l = nullptr;

int choice, selectList;

do {

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.";

continue;

}

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\n12. 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: continue; // Reselecting list handled above

case 12: l1.concatList(l2); break;

case 13: l->reverseDisplay(); break;

case 0: cout << "\n=============== GOOD BYE ===============\n"; break;

default: cout << "Wrong choice"; break;

}

} while (choice != 0);

} while (true);

return 0;

}

**OUTPUT:-**

**Select List**

**1. List 1**

**2. List 2**

**Enter choice: 1**

**1. Create**

**2. Insert President**

**3. Insert Secretary**

**4. Insert After Position (member)**

**5. Display List**

**6. Delete President**

**7. Delete Secretary**

**8. Delete Member**

**9. Find Total No. of Members**

**10. Sort List**

**11. Reselect List**

**12. Combine Lists**

**13. Reverse Display**

**0. Exit**

**Enter your choice: 1**

**Enter PRN number: 101**

**Enter name: Alice**

**=============== List Created ===============**

**1. Create**

**2. Insert President**

**3. Insert Secretary**

**4. Insert After Position (member)**

**5. Display List**

**6. Delete President**

**7. Delete Secretary**

**8. Delete Member**

**9. Find Total No. of Members**

**10. Sort List**

**11. Reselect List**

**12. Combine Lists**

**13. Reverse Display**

**0. Exit**

**Enter your choice: 5**

**=============== List: ===============**

**101 Alice**

**1. Create**

**2. Insert President**

**3. Insert Secretary**

**4. Insert After Position (member)**

**5. Display List**

**6. Delete President**

**7. Delete Secretary**

**8. Delete Member**

**9. Find Total No. of Members**

**10. Sort List**

**11. Reselect List**

**12. Combine Lists**

**13. Reverse Display**

**0. Exit**

**Enter your choice: 0**

**=============== GOOD BYE ===============**