**Assignment- B07**

**Name of Student: Sumit Bhamare**

**Roll No.:08**

**Problem Statement:**

# 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 exist for two divisions. Concatenate two lists.

**Program:**

#include<iostream>

#include<string.h>

using namespace std;

#define TRUE 1

#define FALSE 0

#define SIZE 20

struct node {

  int PRN;

  char name[SIZE];

  struct node \*next;

}\*head;

class sll {

  public:

  sll();

  struct node \*create();

  void display(struct node \*);

  void total();

  struct node \*insert\_President();

  void insert\_Member();

  void insert\_Secretary();

  node \*remove();

  node \*concat(struct node \*, struct node \*);

~sll();

};

  sll::sll() { //constructor defined

  head = NULL;//initialize head to NULL

}

  sll::~sll() { //destructor defined

  node \*temp, \*temp1;

  temp = head->next;

  delete head;

  while (temp != NULL) { //free the memory allocated

  temp1 = temp->next;

  delete temp;

  temp = temp1;

}

}

node \*sll::create() {

 node \*temp=NULL, \*New;

 int val, flag;

 char n[SIZE];

 char ans = 'y';

 flag = TRUE;

 do {

 cout << "\nEnter the PRN of Student: ";

 cin >> val;

 // allocate memory to new node

 cout << "\n Enter Name of Student: ";

 cin >> n;

 New = new node; // allocate memory for new node

 if (New == NULL)

 cout << "Unable to allocate memory\n";

 New-> PRN = val;

 strcpy(New->name, n);

 New-> next = NULL;

 if (flag == TRUE) { // Executed only for the first time

 head = New;

 temp = head;

 flag = FALSE;

} else {

/\*temp last keeps track of the most recently created node\*/

 temp->next = New;

 temp = New;

}

 cout << "\n Do you want to enter more elements? (Y/N) : ";

 cin >> ans;

} while (ans == 'y' || ans == 'Y');

 return head;

}

void sll::display(node \*head) {

 struct node \*temp;

 temp = head;

if (temp == NULL) {

 cout << "\nThe list is empty\n";

return;

}

while (temp != NULL) {

 cout <<"["<< temp->PRN << ","<<temp->name<<"]";

 temp = temp -> next;

}

}

void sll::total() {

 node \*temp;

 int count = 0;

 temp = head;

 if (temp == NULL) {

  cout << "\nThe list is empty\n";

 return;

}

while (temp != NULL) {

 count++;

 temp = temp->next;

}

cout << "\n Total number of members in a club are " << count;

}

node \*sll::remove() {

  node \*temp, \*prev;

  int key;

  prev = new node;

  temp = head;

  cout << "\nEnter the PRN of the node you want to delete: ";

  cin >> key;

  while (temp != NULL) {

  if (temp->PRN == key)//traverse till required node to delete

   break; //is found

  prev = temp;

  temp = temp->next;

}

if (temp == NULL)

  cout << "\nNode not found";

else {

if (temp == head) //first node

  head = temp->next;

else

  prev->next = temp->next; //intermediate or end node

  delete temp;

  cout << "\nThe member is deleted\n";

}

 return head;

}

  void sll::insert\_Secretary() {

  node \*New, \*temp;

  New=new node;

  cout << "\nEnter The PRN of the Student: ";

  cin >> New->PRN;

  cout << "\nEnter The name of the Student: ";

  cin >> New->name;

  if (head == NULL)

  head = New;

  else {

  temp = head;

  while (temp->next != NULL)

  temp = temp->next;

  temp->next = New;

  New->next = NULL;

}

 cout << "\nThe member is inserted\n";

}

void sll::insert\_Member() {

 int key;

 node \*temp, \*New;

 New = new node;

 cout << "\n Enter The PRN of the Student: ";

 cin >> New->PRN;

 cout << "\n Enter The name of the Student: ";

 cin >> New->name;

 if (head == NULL) {

 head = New;

} else {

 cout << "\n Enter The PRN after which you want to insert the node : ";

 cin >> key;

 temp = head;

do {

if (temp->PRN == key) {

 New->next = temp->next;

 temp->next = New;

break;

} else

 temp = temp->next;

} while (temp != NULL);

}

 cout << "\nThe member is inserted\n";

}

 node \*sll::insert\_President() {

 node \*New, \*temp;

 New = new node;

 cout << "\n Enter The PRN of the Student: ";

 cin >> New->PRN;

 cout << "\n Enter the name of the Student: ";

 cin >> New->name;

if (head == NULL)

 head = New;

else {

 temp = head;

 New->next = temp;

 head = New;

}

 cout << "\nThe member is inserted\n";

 return head;

}

 node \*sll::concat(node \*head1, node \*head2) {

 node \*temp;

 temp = head1;

while (temp->next != NULL)

 temp = temp->next;

 temp->next = head2;

 cout << "\n The lists are concatenated";

 return head1;

}

int main() {

sll s;

int choice, ch1;

char ans = 'y';

node \*start=NULL;

node \*start1, \*start2;

start1 = NULL;

start2 = NULL;

do {

  cout << "\n\n1.Create Members";

  cout << "\n2.Display Members";

  cout << "\n3.Insert Member";

  cout << "\n4.Delete Member";

  cout << "\n5.Total Number of Members of Club";

  cout << "\n6.Concatenate two Lists";

  cout << "\n7.Quit";

  cout << "\nEnter Your Choice ( 1-8): ";

  cin >> choice;

 switch (choice) {

 case 1:

  start=s.create();

  break;

 case 2:

  s.display(start);

  break;

 case 3:

  cout << "\nThe Members are \n";

  s.display(start);

  cout << "\nMenu";

  cout << "\n1.Insert President";

  cout<< "\n2.Insert Member";

  cout << "\n3.Insert Secretary";

  cout << "\nEnter your choice: ";

  cin >> ch1;

switch (ch1) {

        case 1:

           start=s.insert\_President();

           break;

        case 2:

           s.insert\_Member();

           break;

        case 3:

           s.insert\_Secretary();

           break;

        default:

           cout << "\nInvalid choice";

        }

        break;

     case 4:

        start=s.remove();

        break;

     case 5:

        s.total();

        break;

     case 6:

        cout << "\n Enter the data for first division\n";

        start1=s.create();

        cout << "\n Enter the data for second division\n";

        start2 = s.create();

        start=s.concat(start1, start2);

        break;

     case 7:

          cout<<"\nExited";

          break;

    default:

        cout << "\nInvalid choice";

    }

  } while (ans != 0);

  return 0;

}

**Output:**

