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.h>
using namespace std;
 struct node
{    int prn,rollno;
     char name[50];
     struct node *next;
};
class info
{       node
*s=NULL,*head1=NULL,*temp1=NULL,*head2=NULL,*temp2=NULL,*head=NULL,*temp=NULL;
       int b,c,i,j,ct;
```

```
        char a[20];
        public:

           node *create();
           void insertp();
           void insertm();
           void delm();
           void delp();
           void dels();
           void display();
           void count();
           void reverse();
           void rev(node *p);
           void concat();

} ;
node *info::create()
{   node *p=new(struct node);
    cout<<"enter name of student ";
    cin>>a;
    strcpy(p->name,a);
      cout<<"\n  enter prn no. of student \n";
    cin>>b;
    p->prn=b;
    cout<<"enter student rollno";
    cin>>c;
    p->rollno=c;
```

```
  p->next=NULL;
  return  p;
}
void info::insertm()
{
    node *p=create();

  if(head==NULL)
  {   head=p;
  }
 else
 {     temp=head;
       while(temp->next!=NULL)
       {   temp=temp->next;   }
          temp->next=p;
  }

 }
  void info::insertp()
{
    node *p=create();

  if(head==NULL)
  {   head=p;
  }
 else
 {     temp=head;
```

```
        head=p;
         head->next=temp->next;

  }

}

 void info::display()
{        if(head==NULL)
         {   cout<<"linklist is empty";
          }
         else
         {
          temp=head;
                cout<<"    prn    rolln0   NAME  \n";
                while(temp!=NULL)
                {    cout<<"   \n"<<temp->prn<<"   "<<temp->rollno<<"   "<<temp->name;
                    temp=temp->next;
                }
               // cout<<"   "<<temp->prn<<"   "<<temp->rollno<<"   "<<temp->name;
         }
}
void info::delm()
{  int m,f=0;
   cout<<"\n enter the prn no. of student whose data you want to delete";
    cin>>m;
    temp=head;
```

```
   while(temp->next!=NULL)
   {
       if(temp->prn==m)
       {         s->next=temp->next;
                 delete(temp);       f=1;
       }
       s=temp;
       temp=temp->next;
    }     if(f==0)
        {   cout<<"\n sorry memeber not deleted ";   }
 }
 void info::delp()
{    temp=head;
   head=head->next;
     delete(temp);
   }
   void info::dels()
{
   temp=head;
   while(temp->next!=NULL)
   {   s=temp;
   temp=temp->next;
   }     s->next=NULL;
     delete(temp);

 }
 void info::count()
```

```
{     temp=head;    ct=0;
      while(temp->next!=NULL)
      {    temp=temp->next; ct++;   }
        ct++;
        cout<<"  Count of members is:"<<ct;

 }
 void info::reverse()
 {    rev(head);   }
 void info::rev(node *temp)
 {    if(temp==NULL)
      { return;   }
      else
      {   rev(temp->next); }
      cout<<"    "<<temp->prn<<"    "<<temp->rollno<<"    "<<temp->name;
 }

 void info::concat()
{  int k,j;
   cout<<"enter no. of members in list1";
   cin>>k;
    head=NULL;
   for(i=0;i<k;i++)
   { insertm();
     head1=head;

   } head=NULL;
```

```
   cout<<"enter no. of members in list2";
    cin>>j;
    for(i=0;i<j;i++)
    { insertm();
      head2=head;

    } head=NULL;

     temp1=head1;
   while(temp1->next!=NULL)
   {   temp1=temp1->next;   }
        temp1->next=head2;


                    temp2=head1;
                   cout<<"    prn    rolln0   NAME  \n";
                  while(temp2->next!=NULL)
                  {
                    cout<<"\n   "<<temp2->prn<<"   "<<temp2->rollno<<"     "<<temp2->name<<"\n";;
                    temp2=temp2->next;
                  }
                    cout<<"\n   "<<temp2->prn<<"   "<<temp2->rollno<<"     "<<temp2->name<<"\n";
  }
int main()
{ info s;
int i;
```

```
  char ch;
do{
  cout<<"\n choice the options";
  cout<<"\n  1. To insert president   ";
  cout<<"\n  2. To insert member   ";
  cout<<"\n  3. To insert secretary ";
  cout<<"\n  4. To delete president   ";
  cout<<"\n  5. To delete member  ";
  cout<<"\n  6. To delete secretary ";
  cout<<"\n  7. To display data   ";
  cout<<"\n  8. Count of members";
  cout<<"\n  9. To display reverse of string ";
  cout<<"\n  10.To concatenate two strings ";
  cin>>i;
 switch(i)
 {      case 1:   s.insertp();
                  break;
        case 2:   s.insertm();
                  break;
        case 3:   s.insertm();
                  break;
        case 4:   s.delp();
                  break;
        case 5:   s.delm();
                  break;
        case 6:   s.dels();
                  break;
```

```
            case 7:   s.display();
                      break;
            case 8:   s.count();
                      break;
            case 9:   s.reverse();
                      break;
            case 10:  s.concat();

                      break;
            default:  cout<<"\n unknown choice";
        }
          cout<<"\n do you want to continue enter y/Y \n";
          cin>>ch;

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

  return 0;
}
```

```
//////////////////
// Online C++ compiler to run C++ program online
#include <iostream>
struct node
```

```
{
   int prn;
   char name;
   struct node *next;
};
struct node *head,*ptr;
void create(int prn, char name)
{
   struct node *ptr = (struct node *)malloc(sizeof(struct node *));
      if(ptr == NULL)
      {
        std::cout<<"\nOVERFLOW\n";
      }
      else
      {
         ptr->prn=prn;
         ptr->name=name;
         ptr->next=NULL;
      }
}
void beginsert(int prn, char name)
 {  create(prn,name);
   ptr->next = head;
   head = ptr;
   std::cout<<"\nNode inserted\n";
 }
```

```
int main ()
{
   int choice,item;
   int prn;
   char name;
   do
   {
      std::cout<<"\nEnter the prn and name\n";
      std::cin>>prn>>name;
      beginsert(prn,name);
      std::cout<<"\nPress 0 to insert more ?\n";
      std::cin>>choice;
   }while(choice == 0);
   return 0;
}
```

```
#include<iostream>
#include<string.h>
using namespace std;
 struct node
{   int prn;
    char name[50];
    struct node *next;
};
class info
```

```
{       node
*s=NULL,*head1=NULL,*temp1=NULL,*head2=NULL,*temp2=NULL,*head=NULL,*temp=NULL;
       int b,c,i,j,ct;
       char a[20];
       public:
          node *create();
          void insertp();
          void inserts();
          void delm();
          void delp();
          void dels();
          void display();
          void count();
          void reverse();
          void rev(node *p);
          void concat();


} ;
node *info::create()//create a new node
{   node *p=new(struct node);
    cout<<"enter name of student ";
    cin>>a;
    strcpy(p->name,a);
      cout<<"\n  enter prn no. of student \n";
    cin>>b;
    p->prn=b;
    p->next=NULL;
    return  p;
```

```
}
void info::inserts()// insert secretary
{
    node *p=create();

   if(head==NULL)
   {   head=p;
   }
  else
  {     temp=head;
        while(temp->next!=NULL)
        {   temp=temp->next;   }
           temp->next=p;
   }

 }
//////////////////////////////////////
 void info::insertm()// insert secretary
 {
     node *p=create();

   if(head==NULL)
   {   head=p;
   }
  else
  {     temp=head;
        while(temp->next!=NULL)
```

```
      {   temp=temp->next;   }
          temp->next=p;
    }

  }

////////////////////////////////////
    void info::insertp()//insert president
 {
     node *p=create();

    if(head==NULL)
    {   head=p;
    }
   else
   {     temp=head;
         head=p;
          head->next=temp->next;

    }

  }

  void info::display()
  {        if(head==NULL)
```

```
        {    cout<<"linklist is empty";
         }
        else
        {
         temp=head;
                cout<<"    prn    NAME   \n";
                while(temp!=NULL)
                {    cout<<"   \n"<<temp->prn<<"   "<<"   "<<temp->name;
                    temp=temp->next;
                }
               // cout<<"   "<<temp->prn<<"   "<<temp->rollno<<"   "<<temp->name;
         }
}
void info::delm()
{  int m,f=0;
   cout<<"\n enter the prn no. of student whose data you want to delete";
    cin>>m;
    temp=head;
    while(temp->next!=NULL)
    {
       if(temp->prn==m)
        {         s->next=temp->next;
                  delete(temp);       f=1;
        }
        s=temp;
        temp=temp->next;
    }     if(f==0)
```

```
        {   cout<<"\n sorry memeber not deleted ";   }
 }
 void info::delp()
{    temp=head;
   head=head->next;
     delete(temp);
   }
   void info::dels()
{
   temp=head;
   while(temp->next!=NULL)
   {   s=temp;
   temp=temp->next;
   }    s->next=NULL;
     delete(temp);

 }
 void info::count()
{     temp=head;    ct=0;
      while(temp->next!=NULL)
      {    temp=temp->next; ct++;   }
        ct++;
        cout<<"  Count of members is:"<<ct;

  }
  void info::reverse()
  {    rev(head);   }
```

```
void info::rev(node *temp)
{    if(temp==NULL)
     { return;   }
     else
     {   rev(temp->next); }
     cout<<"    "<<temp->prn<<"    "<<"    "<<temp->name;
}

void info::concat()
{  int k,j;
   cout<<"enter no. of members in list1";
   cin>>k;
    head=NULL;
   for(i=0;i<k;i++)
   { insertm();
     head1=head;

   } head=NULL;
  cout<<"enter no. of members in list2";
   cin>>j;
   for(i=0;i<j;i++)
   { insertm();
     head2=head;

   } head=NULL;

    temp1=head1;
```

```
   while(temp1->next!=NULL)
   {   temp1=temp1->next;   }
       temp1->next=head2;



                    temp2=head1;
                   cout<<"    prn    rolln0   NAME   \n";
                  while(temp2->next!=NULL)
                  {
                   cout<<"\n   "<<temp2->prn<<"   "<<temp2->name<<"\n";;
                   temp2=temp2->next;
                  }
                   cout<<"\n   "<<temp2->prn<<"   "<<"    "<<temp2->name<<"\n";
  }
int main()
{ info s;
int i;


      char ch;
    do{
      cout<<"\n choice the options";
      cout<<"\n  1. To insert president   ";
      cout<<"\n  2. To insert member   ";
      cout<<"\n  3. To insert secretary ";
      cout<<"\n  4. To delete president   ";
      cout<<"\n  5. To delete member  ";
      cout<<"\n  6. To delete secretary ";
```

```
 cout<<"\n  7. To display data   ";
 cout<<"\n  8. Count of members";
 cout<<"\n  9. To display reverse of string ";
 cout<<"\n  10.To concatenate two strings ";
 cin>>i;
switch(i)
{       case 1:   s.insertp();
                  break;
        case 2:   s.inserts();
                  break;
        case 3:   s.insertm();
                  break;
        case 4:   s.delp();
                  break;
        case 5:   s.delm();
                  break;
        case 6:   s.dels();
                  break;
        case 7:   s.display();
                  break;
        case 8:   s.count();
                  break;
        case 9:   s.reverse();
                  break;
        case 10:  s.concat();

                  break;
```

```
            default:  cout<<"\n unknown choice";
        }
         cout<<"\n do you want to continue enter y/Y \n";
         cin>>ch;

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

  return 0;
 }
#include<iostream>
using namespace std;

 struct node
{   int data;
    struct node *next;
}*head,*con,*head1;
node *create()//create a new node
{
   node *p=new(struct node);
    int a;
    cout<<"enter data ";
    cin>>a;
    p->data=a;
    p->next=NULL;
   return p;
 }
////////////////////////////////////
```

```
void insertFront(){

    node* new_node = new node();
    int data;
    cout<<"enter data ";
    cin>>data;
    new_node->data = data;

    // change the next node of this newNode
    // to current head of Linked List
    new_node->next =head;

    // new_node should become the new head of Linked List
    head = new_node;

    cout << "Inserted Item: " << new_node->data << endl;
}

void insertFront1(){

    node* new_node = new node();
    int data;
    cout<<"enter data ";
    cin>>data;
    new_node->data = data;

    // change the next node of this newNode
```

```
    // to current head of Linked List
    new_node->next =head1;

    // new_node should become the new head of Linked List
    head1 = new_node;

    cout << "Inserted Item: " << new_node->data << endl;
}

void Traverse()
{
    node* temp = new node();
  if(head==NULL)
 {    cout<<"\n linklist is empty";
    }
 else
 {
  temp=head;
  cout<<"\n data in a list are:";
 while(temp!=NULL)
  {
   cout<<"-->"<<temp->data;
   temp=temp->next;
  }
 }
  cout << endl;
}
```

```
//////////insert at end
void insertEnd()
{
   node* new_node = new node();
   node* temp = new node();
   int data;
   cout<<"enter data ";
   cin>>data;
   new_node->data = data;
   new_node->next = NULL;
   if(head==NULL)
   {
      head = new_node;
      cout << new_node->data << " inserted" << endl;
      return;
   }
    temp=head;
   while(temp->next!=NULL)
      temp = temp->next;

   // assign last node's next to this freshNode
   temp->next = new_node;
   cout << new_node->data << " inserted";
}
//////////////////////
void insertPosition (int position)
{
```

```
node *new_node = new node ();
node *temp = new node ();
int data,i;
cout<<"enter data ";
cin>>data;
new_node->data = data;
new_node->next = NULL;

 if(head==NULL)
 {
   new_node->next = head;
   head = new_node;
 }

else
 {
  temp=head;

   // traverse till the current (pos-1)th node
   for(i=1;i<position;i++)
          {
            temp = temp->next;
          }
   new_node->next = temp->next;
   temp->next = new_node;
 }
```

```
}
void deleteFront()
{
node *ptr = new node ();
 if(head==NULL)
  cout<<"list is empty";
 else
 ptr=head;
 head=head->next;
 cout<<"deleted element is"<<ptr->data;
 delete ptr;
}
//////////////////
void deleteLast(){
    node *temp = new node ();
    node *ptr = new node ();
    if(head==NULL)
    cout<<"list is empty”;
   else
   if (head->next==NULL)
 {
   ptr=head;
   head=NULL;
   cout<<"element deleted is"<<ptr->data;
   delete ptr;
 }
 else
```

```
{
   ptr=head;
   while(ptr->next!=NULL){
   temp=ptr;
   ptr=ptr->next;
   cout<<"hello"<<ptr->data;
   }
   temp->next=NULL;
   cout<<"element deleted is"<<ptr->data;
   delete ptr;
 }
}
//////////////////////
void deletePosition(int loc){
   int i;
   node *temp = new node ();
   node *ptr = new node ();
   if(head==NULL)
      cout<<"list is empty";
   else
    ptr=head;
  for(i=0;i<loc;i++){
   temp=ptr;
   ptr=ptr->next;
   i=i+1;}
 cout<<"element is deleted"<<ptr->data;
 temp->next=ptr->next;
```

```
delete ptr;
}
////////////////////////////
int search(int key)
{   int i=1;
   node *ptr = new node ();
   ptr = head;
  if(ptr == NULL)
  cout<<"EMPTY LIST";
  else
  while(ptr!=NULL){
      if(ptr->data==key)
      return i;
      else
       i++;
       ptr=ptr->next;
  }
  return -1;
}
////////////////////////
void sort()
{
    node *temp = new node ();
    node *min = new node ();
    node *r = new node ();
    temp=head;
    while (temp) {
```

```
        min = temp;
        r = temp->next;

        // Traverse the unsorted sublist
        while (r) {
           if (min->data > r->data)
               min = r;

           r = r->next;
        }

        // Swap Data
        int x = temp->data;
        temp->data = min->data;
        min->data = x;
        temp = temp->next;
    }
}
node * concatenate (struct node *head1, struct node *head2)
{
    node *p= new node();
    if (head1==NULL)                    return  (head2);
    if (head2==NULL)                    return (head1);
    p=head1;
    while (p->next!=NULL)
        p=p->next;
        p->next=head2;
```

```
return (head1);
}
int main()
{
 insertFront();
 Traverse();
 //insertFront1();
 //Traverse();
 insertEnd();
 Traverse();
  //insertPosition(2);
 //deleteFront();
 //deleteLast();
 //deletePosition(1);
  //Traverse();
  //cout<<search(13);
  //sort();
  con=concatenate(head,head1);
  Traverse();
   return 0;
}
```