Assignment No. 7

Title:

Write a C++ program to create pinnacle club data base using Singly Linked List.

Objectives::

1)To study the dynamic memory allocation.

2)To understand Singly Linked List.

3)To study the concept of concatenation of two singly linked lists.

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 exists for two divisions. Concatenate two lists.

Software Requirements::

- ✓ Ubuntu with Linux 14.04,GCC / G++ (Editor).

Hardware Requirements::

- ✓ Any Processor above Pentium 4.

Theory & Concept::

1) Dynamic Memory Allocation ::

In case of dynamic data structures, the memory space required variables is calculated and allocated during execution. Dynamic memory in managed in ‘C++’ through a set of library function.

- ✓ **Allocating a block of memory in "C++"**

ptr =(cast-type)new(byte-size); The "new ()" returns a pointer (of cast type) to an area of memory with

size, byte-size

- ✓ **Example::**x=(int*)new(100*sizeof(int));

2) Singly Linked List ::

In this type of linked list two successive nodes of the linked list are linked with each other in sequential

linear manner. Movement in forward direction is possible

a) Create Function

Assume n=3 (3 nodes to be created) with inputs as 5,1,9. The address of the newly acquired node pointed

by P. Subsequently, P is moved to the next node.

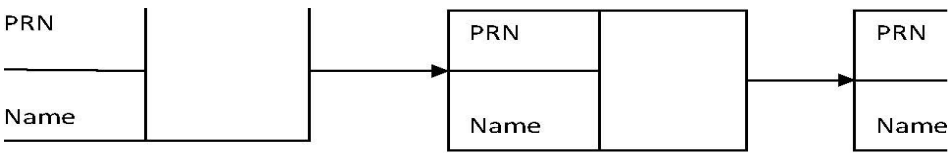
E.g.

```
#include <iostream.h>
typedef struct node
{
        int data; struct node *next;
}node; node*
Create node ()
{
        node*head,*p,*q;
        int i,no; head=NULL;
        cout <<"\nEnter the no of nodes::"; cin >>"%d",&no;
                for(i=0;i <no;i++)
                {
                        p=(node*)malloc(sizeof(node));
                        cout<<"\nEnter the data::";
                        cin >>"%d",&(p->data; p->next=NULL;
                        if(head==NULL)
                                head=p;
```

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

b) Print Function

```
void print (node*head)
{
        node*p;
        int count=0;
        cout<<"\nList of elements are ::";
        p=head;
        while(p!=NULL)
        {
                count++;
                cout<<"%d=>",p->data;
                p=p->next;
        }
        Cout<<"\nTotal no of elements=%d",count;
}
```



c) Insert at beginning

Algorithm

1. Obtain space for new node.
2. Assign data to the data field of the new node.
3. Set the next field of the new node to the beginning of the linked list.
4. Change the reference pointer of the linked list to point to the new node.

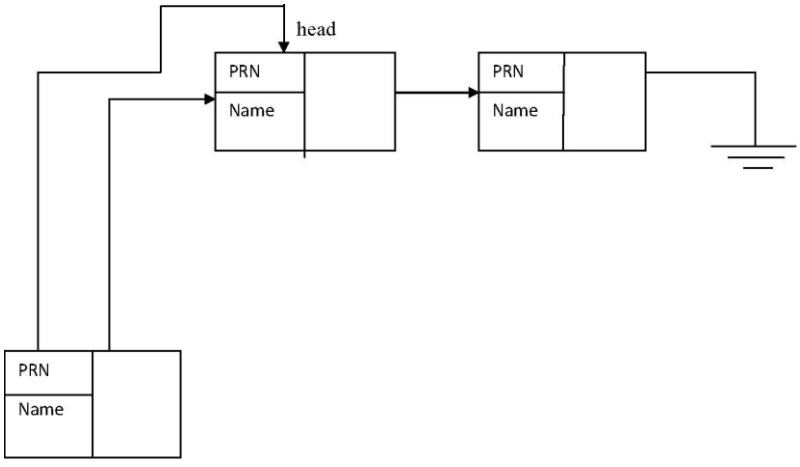
E.g.

```
node*insertb (node*head,int ele)
{
        node *p;
        p=(node*)malloc(sizeof (node));
        p->data=ele; p->next=NULL;
        if(head==NULL)
        {
                head=p;
                return head;
                }
                else
                {
                        p->next=head;

                        head=p;

                        return head;

                }
```



d) Insert at location

Algorithm

1.Acquire memory for new node with its address in pointer P.

i.e.P = (node *) malloc(sizeof(node));

2.Assign value to data field and make its ‘next’ field ‘NULL’.

i.e.P □ data =x; P □ next = NULL;

3.if (LOC == 1)

Then insert the node, pointed by P, at the beginning of the linked list. This can be done by following

steps.

(a)Store head in the next field of the node pointed by P

1 P ->next = head;

(b)Move head to the newly connected node.

2 head = P; go to step 7 4.[If Loc > 1] Position a pointer q on (LOC -1)th node .

3 q = head; for (i = 1; i < (Loc -1); i++) q = q-> next;

5. if q is NULL then report “overflow” and terminate the algorithm. Go to step 7
6. Insert the node pointed by P, after the node pointed by q.

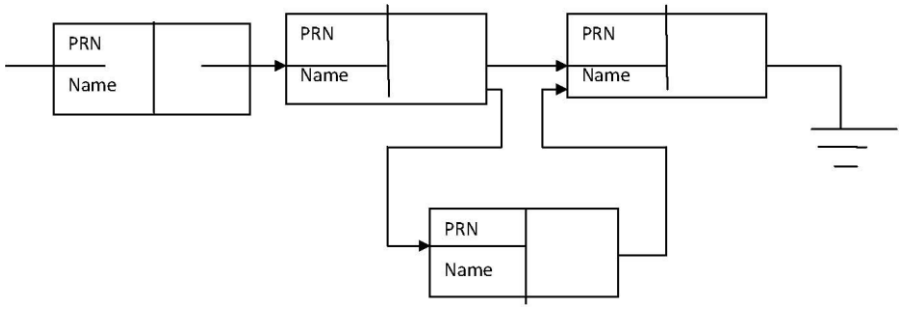
4 P ->next = q->next; q ->next = p;

7.Stop.

E.g.

```
node *insertl(node*head,int ele,int loc)
{
        node *p,*q; int i; p=(node*)malloc(sizeof(node)); p->data=ele;
        p->next=NULL; if (loc==1)
        {
        p->next=head; head=p;
        return head;
        }
        q=head; for (i=1;i <loc-1;i++)
```

```
if (q->next!=NULL)
        q=q->next;
else
{
        cout << "\nOverflow"; return
        head;
}
p->next=q->next;
q->next=p;
return head;
}
```



e) Insert at end Algorithm

Insert an item 'x' in a linked list, referenced by the pointer 'head'

1. Acquire memory for new node

        i.e.p = (node*) malloc(sizeof (node);

2. Assign value to the data filed and make its 'next' field 'NULL'

        i.e.P □ data = x; P □ next = NULL;

3. If 'head' is 'NULL' Then head = p ; goto step 6

4. Position a pointer q on the last node by traversing the linked list form the first node and until it reaches the last node. i.e. q= head while (q □ next ! = NULL) q= q □ next ;

5. Store the address of the newly acquired node, pointed by P, in the next field of node pointed by q.

i.e. q □ next = p;

6. Stop.

E.g.

```
node*inserte(node*head,int ele)
{
        node *p,*q; p=(node*)malloc(sizeof (node));
        p->data=ele;
        p->next=NULL;
if(head==NULL)
{
        head=p; return head;
}
for(q=head;q->next!=NULL; q=q->next);
        q->next =p;
return head;
}
```

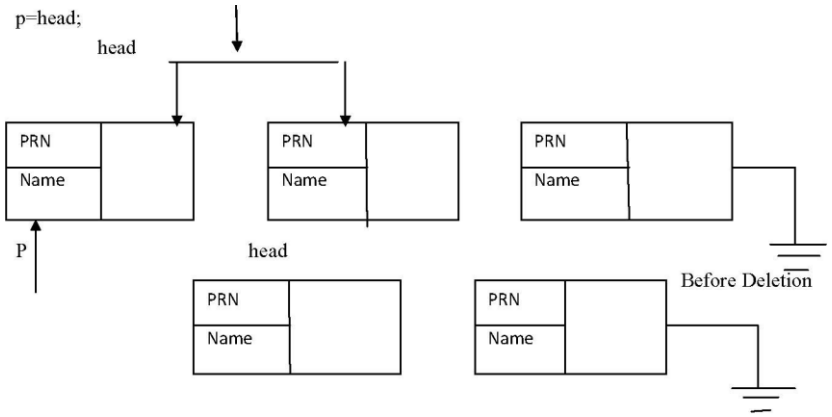
f) Delete at beginning Algorithm

1. Store the address of the first node in a pointer variable, say P.
2. Move the head to the next node.
3. Freethe node whose address is stored in the pointer variable P.

E.g.

```
node *delb(node*head)
{ node *p;
        if(head==NULL)
        {
                cout<< "\nList is empty";
                return head;
        }
After Deletion
head=head->next;

p->next=NULL;
delete p;
return head; }
```



f) Delete at location

Algorithm

1. Store the address of the preceding node in a pointer variable P. Node to be deleted is marked as
   key node.
2. Store the address of the key node in a pointer variable q, so that it can be freed subsequently.
3. Make the successor of the key node as the successor of the node pointed by P.
4. Free then ode whose address is stored in the pointer variable q.

E.g.

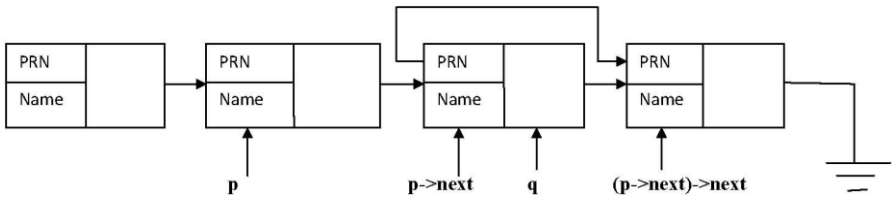
```
node *dell(node *head,int loc)
{
        node*p,*q;
        int i;
                if (loc==1)
                {
                        p=head;
                        head=head->next;
                        p->next=NULL;
                        delete p;
                        return head;
                }
```

```
        q=head;
for (i=1;i<loc-1;i++);
if(q!=NULL)
    q=q->next;
else
{
    Cout<< "\nUnderflow"; return head;
}
p=q->next;
p->next=p->next;
p->next=NULL;

delete p;

return head;

}
```



H) Delete at end Algorithm

[Deleting last node of a linked list, referenced by the pointer head]

In order to delete the last node, we must position a pointer q on the last but one node. Address of the nodeto be deleted is stored in pointer P, So that the memory allocated to it can be freed.

1. If the firstnode itself is the last node then [make the linked list empty]

```
1 if (head □ next == NULL)
{
        free (head); head = NULL;
        goto step 4
```

}

2.[otherwise] position a pointer q on last but one node

2 q = head; while (q->next ->next! = NULL) q = q-> next; 3.Delete the last node

3 p = q->next; free (p);

q ->next = NULL; 4.Stop.

E.g.

```
Node*dele(node*head)
{ node*p,*q;
if(head==NULL)
{
        cout << "\nList is empty"; return head;
}
for (q=head;q->next->next!=NULL;q=q->next); p=q->next; q->next=NULL; delete p;
return head; }
```

i) Concatenate two lists

Algorithm

Let us assume that the two linked lists are referenced by head1 and head2 respectively.

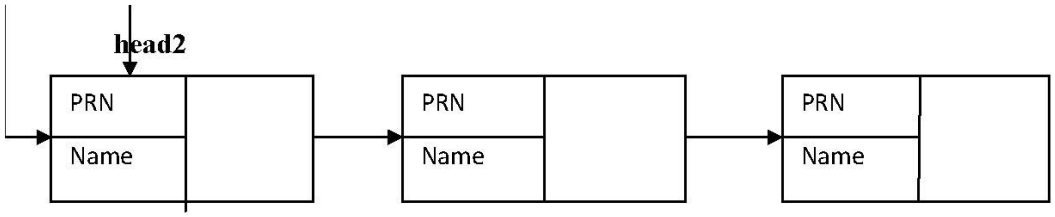
(1)If the first linked list is empty then return head2.

(2)If the second linked list is empty then return head1.

(3)Store the address of the starting node of the first linked list in a pointer variable, say P.

(4)Move the P to the last node of the linked list through simple linked list traversal technique.

(5)Store the address of the first node of the second linked list in the next field of the node pointed by P. Return head1



E.g.

```
node * concatenate(node *head,node *head2) { node *p; if (head1==NULL) return head 2;
if (head2==NULL)
return head 1;
p=head 1;
while(p->next!=NULL)
```

Conclusion::

By using Singly Linked List and concept of concatenation we have successfully implemented the program.

Program :

```
#include<iostream>
using namespace std;
struct node
{
	int prn;
	char name[20];
	node *next;
};
class pinnacle
{
public:
	node *head,*temp;

	pinnacle()
{
		head=NULL;
		temp=NULL;
}
void create();
void display();
void insert_front();
void insert_middle();
void insert_end();
void delete_front();
```

```
void delete_end();
void delete_middle();
void count();
void rev(node *);
};
        void pinnacle::create()
        {
                 char ans;
                node *newnode,*current;
           do
           {
                newnode=new node;
                cout<<"enter a prn number ";
           cin>>newnode->prn;
           cout<<"enter a name ";
                   cin>>newnode->name;
           newnode->next=NULL;
           if(head==NULL)
           {
                head=newnode;
                current=newnode;
           }
           else
           {
           current->next=newnode;
```

```
        current=newnode;
        }

        cout<<"do you want to exit if yes enter y";
        cin>>ans;
        }while(ans=='y');

    }
    void pinnacle::display()
    {
node *p;
cout<<"\n PRN no.\tname \n";
if(head!=NULL)
{
p=head;

while(p!=NULL)
{
    if(p==head)
    {
            cout<<"\n"<<p->prn;
            cout<<"\t"<<p->name;
            cout<<"\t  -> president";
            p=p->next;
    }
```

```
    else if(p->next!=NULL)
    {
            cout<<"\n"<<p->prn;
     cout<<"\t"<<p->name;
    p=p->next;
    }
    else
    {
            cout<<"\n"<<p->prn;
            cout<<"\t"<<p->name;
    cout<<"\t ->  secretory";
                    p=p->next;
    }

}
}
else
{
    cout<<"list is empty";

}
    }
    void pinnacle::insert_front()
    {
            node *temp;
```

```
            temp=new node;
            cout<<"enetr a prn no";
            cin>>temp->prn;
            cout<<"enter name";
            cin>>temp->name;
            temp->next=head;
            head=temp;

    }

    void pinnacle::insert_middle()
    {
            node *temp,*p;
int position;
                        temp=new node;
                        cout<<"enter a  prn number as position ";
                        cin>>position;
                        cout<<"enter a prn no";
                        cin>>temp->prn;
                        cout<<"enter name";
                        cin>>temp->name;
                        p=head;
                        while(p!=NULL)
                    {
                                if(p->prn==position)
```

```
                                        {
                                                break;
                                        }
                                        else
                                        {
                                                p=p->next;
                                        }

                                }
                                temp->next=p->next;
                                p->next=temp;
}
        void pinnacle::insert_end()
        {
    node *temp,*p;
    temp=head;
    p=new node;
    cout<<"enter a prn no";
    cin>>p->prn;
    cout<<"enter a name";
    cin>>p->name;
    p->next=NULL;
    while(temp->next!=NULL)
    {
       if(temp->next==NULL)
```

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


          void pinnacle::delete_front()
          {
     node *p;
     p=head;
     head=p->next;
     delete p;
          }
void pinnacle::delete_middle()
{
          int position;
          node *p,*temp;
          temp=head;
          cout<<"enter prn which you want to dalete";
```

```
        cin>>position;
        while((temp->next)->next!=NULL)
        {
                if((temp->next)->prn==position)
                {
                        break;
                }
                else
                {
                        temp=temp->next;
                }
        }
        p=temp->next;
                                temp->next=p->next;
                                delete p;

}
void pinnacle::delete_end()
{
node *p,*temp;
temp=head;
while((temp->next)->next!=NULL)
{
        temp=temp->next;
}
```

```
p=temp->next;
temp->next=NULL;
delete p;

}

void pinnacle::count()
{
        int i=0;
        node *p;
        p=head;
        while(p!=NULL)
        {

                i++;
                p=p->next;

        }
        cout<<"\n total number of members of club: "<<i;
}

void pinnacle::rev(node *head)
{
        node *p;
        p=head;
```

```
        /*int **a[20];
        int i=0,j=0;
        while(p!=NULL)
        {
                a[i]=&p;
                i++;
                p=p->next;
        }
        for(j=i;j>=0;j--)
        {
                cout<<a[j];
        }*/
        if(p==NULL)
        {
                return;
        }
        else
        {
                rev(p->next);
        }
                cout<<"\n"<<p->prn<<"\t"<<p->name;


        }

int main()
```

```
{
        int ch;
        char ans;
        pinnacle p,q;
        p.create();
        p.display();
do
{
   cout<<"\n \n ------------------------------------MAIN MENU-----------------------------------------";
        cout<<"\n 1.add president";
        cout<<"\n 2.add student.";
        cout<<"\n 3.add secretary.";
        cout<<"\n 4.delete president";
        cout<<"\n 5.delete member";
        cout<<"\n 6.delete secretary";
        cout<<"\n 7.count total no. of members.";
        cout<<"\n 8.display member of club.";
        cout<<"\n 9.reverse the string.";
        cout<<"\n 10.concatinate two string";
        cout<<"\n ------------------------------------------------------------------------------";
        cout<<"\n enter a choice";
        cin>>ch;
        switch(ch)
        {
        case 1:
```

```
            p.insert_front();
            p.display();

            break;
case 2:
            p.insert_middle();
            p.display();
            break;
case 3:
            p.insert_end();
            p.display();
            break;
case 4:
            p.delete_front();
            p.display();
            break;
case 5:
            p.delete_middle();
            p.display();
            break;
case 6:
            p.delete_end();
            p.display();
            break;
case 7:
```

```
                p.count();
                break;
        case 8:
                p.display();
                break;
        case 9:
                p.rev(p.head);
                break;
        case 10:
              q.create();
        p.temp=p.head;
       while(p.temp->next!=NULL)
              {
         p.temp=p.temp->next;
              }
       p.temp->next=q.head;
       p.display();
       break;
        default:
                break;
        }
cout<<"do you want to continue if yes enter y";
cin>>ans;
}while(ans=='y');
        return 0;
```

```
}
```

```
/*
enter a prn number  1
enter a name  qwe
do you want to exit if yes enter y y
enter a prn number   2
enter a name  ert
do you want to exit if yes enter y n

 PRN no.            name

1          qwe        -> president
2          ert        ->  secretory

 ------------------------------------MAIN MENU-----------------------------------------
 1.add president
 2.add student.
 3.add secretary.
 4.delete president
 5.delete member
 6.delete secretary
 7.count total no. of members.
 8.display member of club.
```

9.reverse the string.

10.concatinate two string

---

enter a choice 1

enetr a prn no 3

enter name rt

PRN no. name

3 rt -> president

1 qwe

2 ert -> secretory

do you want to continue if yes enter y y

----------MAIN MENU----------

1.add president

2.add student.

3.add secretary.

4.delete president

5.delete member

6.delete secretary

7.count total no. of members.

8.display member of club.

9.reverse the string.

10.concatinate two string

---

enter a choice 2

enter a prn number as position 1

enter a prn no 5

enter name th

PRN no. name

3 rt -> president

1 qwe

5 th

2 ert -> secretory

do you want to continue if yes enter y y

----------MAIN MENU----------

1.add president

2.add student.

3.add secretary.

4.delete president

5.delete member

6.delete secretary

7.count total no. of members.

8.display member of club.

9.reverse the string.

10.concatinate two string

---

enter a choice 3

enter a prn no 90

enter a name td

PRN no. name

3 rt -> president

1 qwe

5 th

2 ert

90 td -> secretory

do you want to continue if yes enter y y

----------MAIN MENU----------

1.add president

2.add student.

3.add secretary.

4.delete president

5.delete member

6.delete secretary

7.count total no. of members.

8.display member of club.

9.reverse the string.

10.concatinate two string

---

enter a choice 4

PRN no. name

1 qwe -> president

5 th

2 ert

90 td -> secretory

do you want to continue if yes enter y y

----------MAIN MENU----------

1.add president

2.add student.

3.add secretary.

4.delete president

5.delete member

6.delete secretary

7.count total no. of members.

8.display member of club.

9.reverse the string.

10.concatinate two string

---

enter a choice 5

enter prn which you want to dalete 5

PRN no. name

1 qwe -> president

2 ert

90 td -> secretory

do you want to continue if yes enter y y

----------MAIN MENU----------

1.add president

2.add student.

3.add secretary.

4.delete president

5.delete member

6.delete secretary

7.count total no. of members.

8.display member of club.

9.reverse the string.

10.concatinate two string

---

enter a choice 6

PRN no. name

1 qwe -> president

2 ert -> secretory

do you want to continue if yes enter y y

----------MAIN MENU----------

1.add president

2.add student.

3.add secretary.

4.delete president

5.delete member

6.delete secretary

7.count total no. of members.

8.display member of club.

9.reverse the string.

10.concatinate two string

---

enter a choice 7

total number of members of club: 2

do you want to continue if yes enter y y

----------MAIN MENU----------

1.add president

2.add student.

3.add secretary.

4.delete president

5.delete member

6.delete secretary

7.count total no. of members.

8.display member of club.

9.reverse the string.

10.concatinate two string

---

enter a choice 8

PRN no. name

1 qwe -> president

2 ert -> secretory

do you want to continue if yes enter y y

----------MAIN MENU----------

1.add president

2.add student.

3.add secretary.

4.delete president

5.delete member

6.delete secretary

7.count total no. of members.

8.display member of club.

9.reverse the string.

10.concatinate two string

---

enter a choice 9

2 ert

1 qwe

do you want to continue if yes enter y 10

*/