# Experiment No - 2

Aim :-

Department of Artificial Intelligence has Student's club named 'SAAI' Student 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 Secre-tary club. Write program to maintain club's member's information using singly ~~bi~~linked list store Student MIS Registrati-on No & Name. Write function to

a) Add and delete the members as well as president or even Secretary.
b) Compute total Number of member of club.
c) Display members
d) Display list in reverse order using recursion.
e) Two LinKed List exists for two division. Concatenate two list.

Theory :-

A linked list is a sequence of data structure, which are connected together via links. Linked list is a sequence of link which contains item. Each link contains a connection to another link. LinkedList is the second most-used data.

Structure after array. Following are the important terms to understand the concept of Linked List.

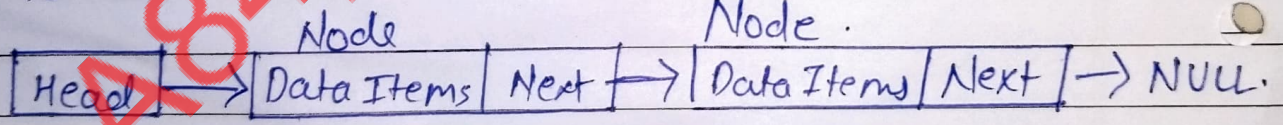
* Link – Each link of a linked list can store a data called an element.

* Next – Each link of a linked list contains a link to the next link called Next.

* LinkedList – A Linked List contains the connection link to the first link called first.

Linked List Representation –

Linked ~~list~~ List can be visualized as a chain of nodes, where every node points to the next node.




As per the above illustration, following are the important points to be considered

- Linked list contain a link element called first
- Each link carries a data fields and a link field called Next.

— Each link is linked with its next link using its next link.
— Last link carries a link as null to mark the end of the list.

Types of linked list :-

Following are the various types of linked list.

* Simple linked list —
  Item navigation is forward only.

* Doubly linked list —
  Item can be navigated forward and backward.

* Circular linked list —
  Last item contains link of the first element as next and the first element has a link to the last element as previous.

Basic Operation :—
* Insertion — Adds an element at the beginning of the list.

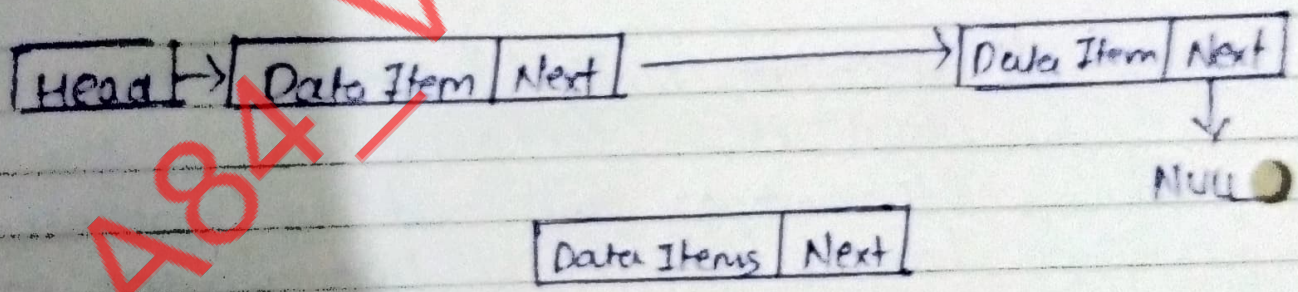
* Deletion — Deletes an element at the beginning of the list.

* Display :- Displays the complete list.

* Search - Searches an element using the given Key.

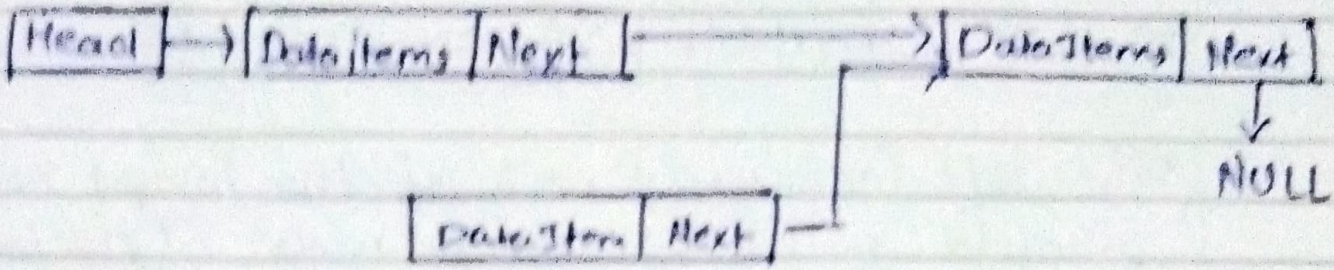
* Delete - Delete an elements using the given key

Insertion Operation :-
Adding a new node in linked list is a more than one step activity. We shall learn this with diagrams here.
First, create a node using the same structure and find the location where it has to be inserted.



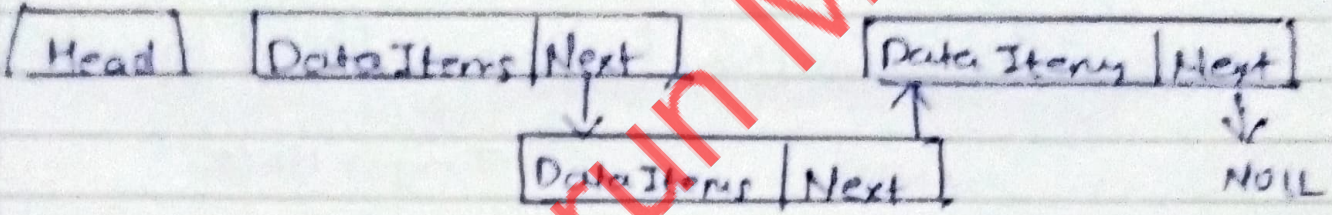

Imagine that we are inserting a node B (New Node) between A (Left Node) and C (Right Node). Then point B.next to C -

NewNode.next → RightNode;

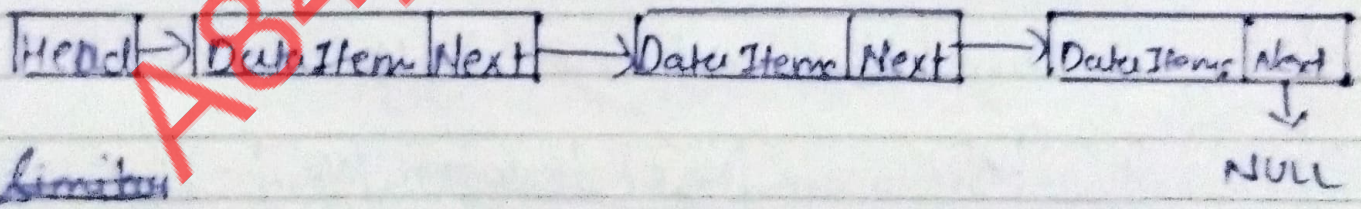



Now, the next node at the left should point to the new node.

Left Node.next → NewNode;



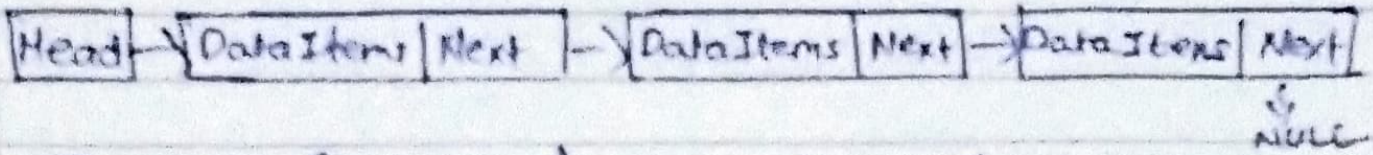

This will put the new node in the middle of two. The new list should look like this




~~Limitation~~

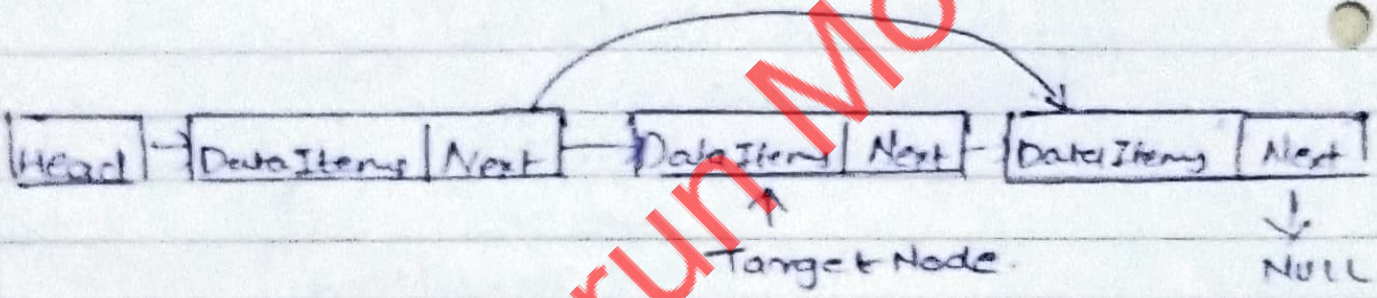
Deletion Operation :-

- Deletion is also a more than one step process. we shall learn with pictorial representation. First locate the target Node to be removed by using Searching algorithms



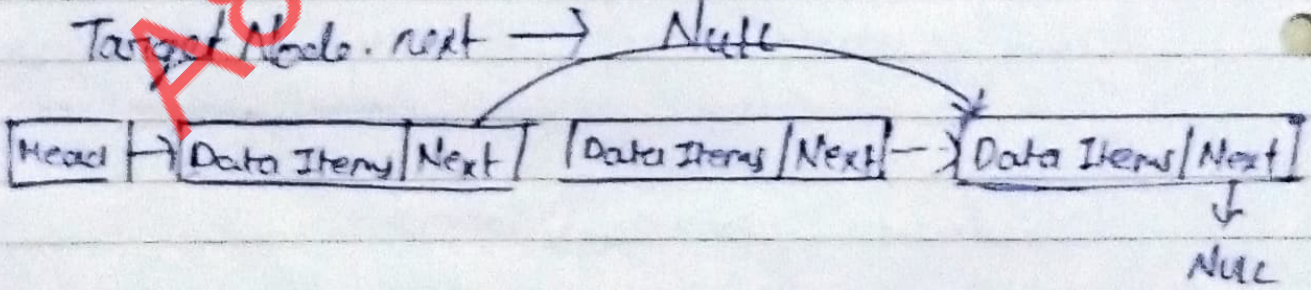

The left (previous) node of the target node now should point to the next node of the target node.

Left Node . Next ⟶ Target Node . Next

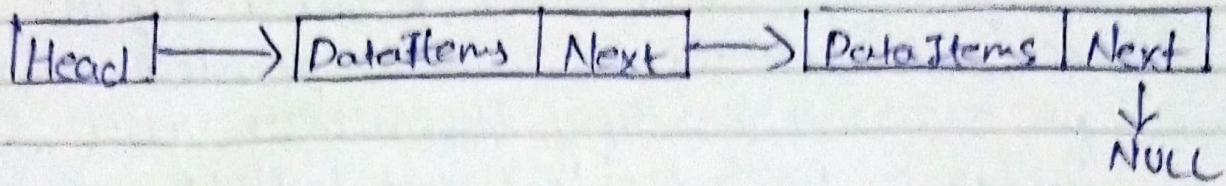



This will remove the link that was pointing to the target node. Now using the code below, we can remove what the target node is pointing at.

Target Node . next ⟶ Null

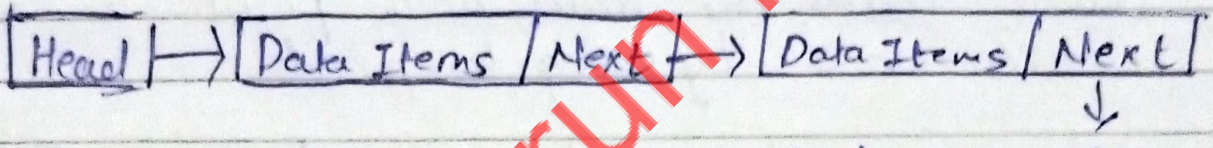



We need to use the deleted node. We can keep that in ~~ram~~ memory otherwise, we can simply deallocate memory and wipes off the target node completely.



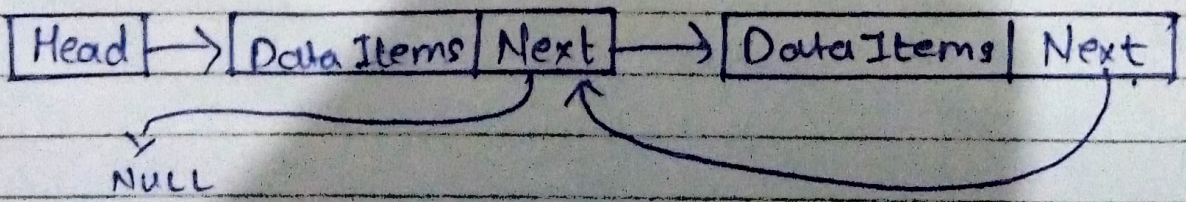

Reverse Operation :—

- This operation is through one. We need to make the last node to be pointed by the head node and reverse the whole linked list.

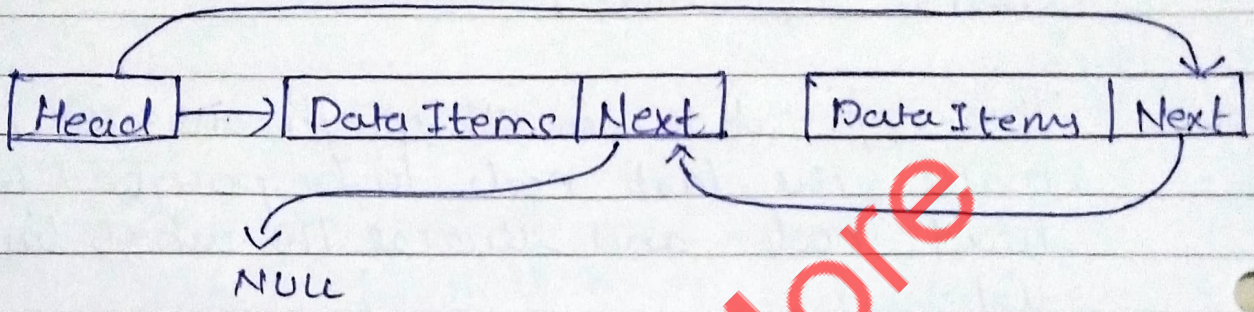



- First we traverse the end of list. It should be pointing to NULL. Now, we shall make it point to the previous node.

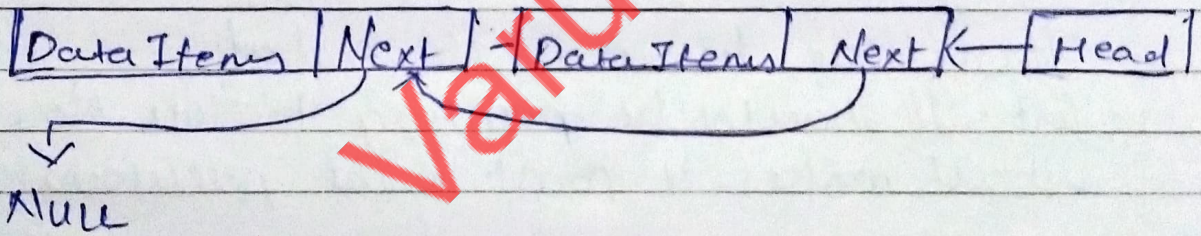
- We have to make sure that last Node is not the last node. So we'll have some temp node, which looks like the head node pointing to the last node Now, we shall make all left side Nodes point to their previous node one by one.



- Except the (first node) pointed by the head node, all nodes should be pointed to their predecessor, making them their new successor. The first node will point to NULL.




we'll make the head node point to the new node by using temp node.




Thus, the linked list is now reversed.

Conclusion :—

This way we implemented, operation on Singly LinkedList.

# Code 2

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

class list;

class node
{
    int MIS;
    string name;
    node *next;

public:
    node(int x, string nm)
    {
        MIS = 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() //function is only for passing start as argument to recursive function
    {
        if (start == NULL)
            return false;
        node *temp = start;
        displayRev(temp);
        //cout<<"(President)";
        return true;
    }
};
void list::displayRev(node *t)
{
    if (t == NULL)
        return;
    else
    {
        displayRev(t->next);
        cout << "\nMIS NO:" << t->MIS << " Name: " << t->name;
    }
}
void list::create()
{
    int no;
    string nam;
    if (start == NULL)
    {
        cout << "Enter MIS 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->MIS << "  " << t->name << " \n";
            t = t->next;
        }
        //cout<<t->MIS<<"  "<<t->name<<" \n";
    }
}
void list::insertAtBeginning()
{
    int no;
    string nam;
    node *temp;
    if (start == NULL)
    {
        create();
    }
    else
    {
        cout << "\nEnter MIS number: ";
        cin >> no;
        cout << "Enter name: ";
        cin >> nam;
        //cout<<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 MIS 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 MIS No. after do you want insert:";
    cin >> prev_no;
    node *t;
    t = start;
    string nam;
    int flag = 0, no;
    while (t != NULL)
    {
        if (t->MIS == prev_no)
        {
            flag = 1;
            break;
        }
        t = t->next;
    }
    if (flag == 1)
    {
        node *p;
        cout << "\nEnter MIS 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 list.";
    }
}

void list::deleteAtFirst()
{
    node *t;
    if (start == NULL)
        cout << "\nClub is Empty..";
    else
    {
        t = start;
        start = start->next;
        t->next = NULL; //Not necessary
        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 MIS no. of member to be deleted: ";
        cin >> no;
        t = start->next; //t=start if we have to delete precident also.. start->next is first member
        while (t->next != NULL)
        {
            if (t->MIS == no)
            {
                flag = 1;
                break;
            }
            prev = t;
            t = t->next;
        }
        if (flag == 1)
        {
            prev->next = t->next;
            t->next = NULL;
            delete t;
            cout << "\nMember with MIS 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 tMIS;
    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->MIS) > (j->next->MIS))
            {
                tMIS = j->MIS;
                tname = j->name;
                j->MIS = j->next->MIS;
                j->name = j->next->name;

                j->next->MIS = tMIS;
                j->next->name = tname;
            }
        }
    }
    cout << "\n List 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; //first list
    while (p->next != NULL)
    {
        p = p->next;
    }
    p->next = t;
    q1.start = NULL; //second list is set to  null
    cout << "\nAfter concatenationlist";
    display();
}
int main()
{
    list *l;
    int choice, selectList;
    list l1, l2;
    l = &l1;
X:
    cout << "Welcome to AI GHRCEM!" << endl;
    cout << "A84 Varun More" << endl;
    cout << "\nSelect List\n1.List 1(Div-1)\n2.List 2(Div-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)\n5.Display list"
             << "\n6.Delete President\n7.Delete Secretary\n8.Delete Member\n9.Find total No. of members\n10.Sort list\n11. Reselect Li
```

```
             << "\n12.Combine lists\n13.Reverse Display\n0. Exit\nENter your choice:\t";
        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;
        deafult:
            cout << "Wrong choice";
        }
    } while (choice != 0);
    cout << "\n========== GOOD BYE ==================\n";

    return 0;
}
```



**Output:**

```
ENter your choice:      1
Enter MIS number: 123
Enter name: Varun
Varun
====== 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:
```

```
0. Exit
ENter your choice:      2

Enter MIS number: 123
Enter name: Varun
Inserted  Varun at the beginning.
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. Exit
ENter your choice:      2

Enter MIS number: 1234
Enter name: Eshan
Inserted  Eshan at the beginning.
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:
```

```
====== List: ======
1234  Eshan
123  Varun
123  Varun

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:
```

```
ENter your choice:      6

President deleted..
```

```
0. Exit
ENter your choice:      7

Secretary Deleted.
1. create
```

```
ENter your choice:      8

Enter MIS no. of member to be deleted: 45

Member not found in List./president or secretary cannot be deleted.
```

```
0. Exit
ENter your choice:      9

Total members(including President & Secretary): 3
```

```
ENter your choice:      10

 List is sorted.
====== List: ======
123  Varun
123  Varun
1234  Eshan
```

```
ENter your choice:      12

After concatenationlist
====== List: ======
123  Varun
123  Varun
1234  Eshan
1456  Pratamesh
```

```
0. Exit
ENter your choice:      13

MIS NO:1234 Name: Eshan
MIS NO:123 Name: Varun
MIS NO:123 Name: Varun
1. create
```