Experiment No: 02

Aim:

Department of Computer Engineering has student's club named 'COMET'. 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 node of president of club and last node is reserved for node of secretary of club. Write a program to maintain club member's information using singly link list. Store students MIS registration No. and Name. Write functions a) to add and delete the members as well as president or even secretary. b) Compute total no. of members of club. c) Display members. d) Display list in reserve order using recursion. e) Two linked list exists for two division. Concatenate two lists.

Theory:

A linked list is a data structure sequence, which can connected together via links. Linked list is a sequence of links which contains items. Each link contains a connections to another link. Linked list is the second most used data structure after array. Following are the important terms to understand the concept of linked list.

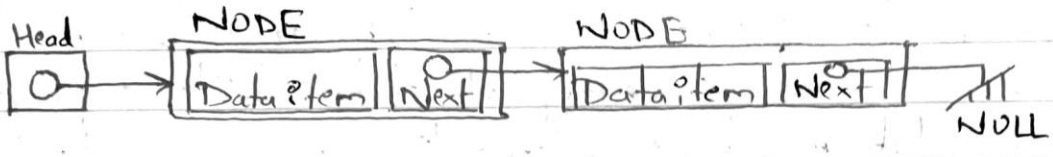
Link - Each link of linked list can store a data known element.

Next - Each link of linked list contain a link to the next link called Next.

Linked list - It contains the connection link to the first link called First.

Linklist Representation:-

Linklist can be visualized as chain of node where every node points to next node.



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

Linklist contains a link element called first.
Each link carries a data field & link field called next.
Each link is linked with its next link using next link.
Last link carries a link as null to mark end of list.

Types of Linklist:-

Simple Linked list - Item navigation is forward only.

Doubly Linked list - Item can be navigated forward & backward.

Circular linked list - Last item contain link of first element as next & ~~last~~ first element has link to last element as previous.

Basic Operations:-

Insertion - Add element at beginning of list.

Deletion - Deletes an element at beginning of list.

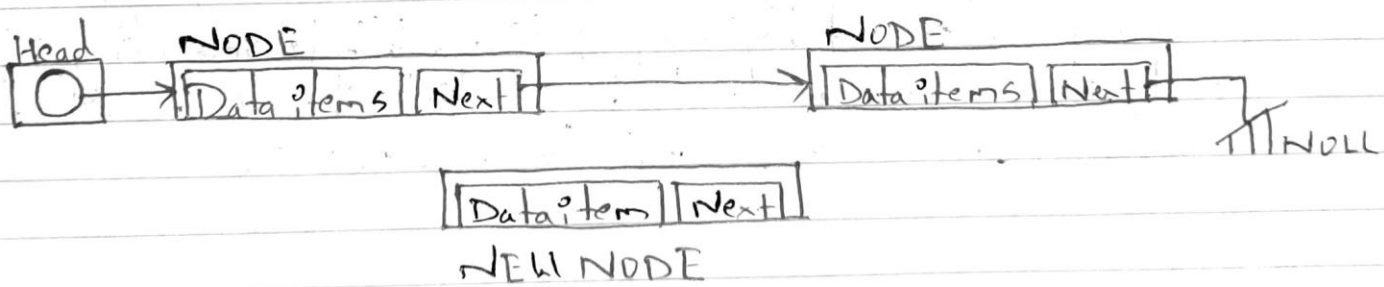
Display - Displays the complete list.

Search - Searches an element using the given key.

Delete - Deletes an element using the given key.

Insertion Operation:-

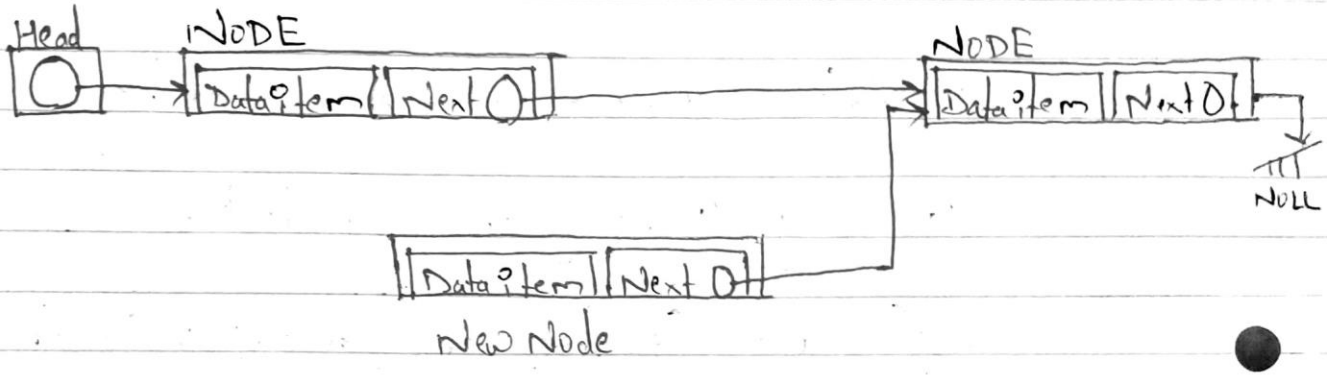
Adding a new node in linked list is more than one step activity. We shall learn this with diagrams here. First, create a node using the same structure and find the location where it has to be inserted.



Imagine that we are inserting node B (new node), between A (Left Node) & C (Right Node). Then point B.next to C -

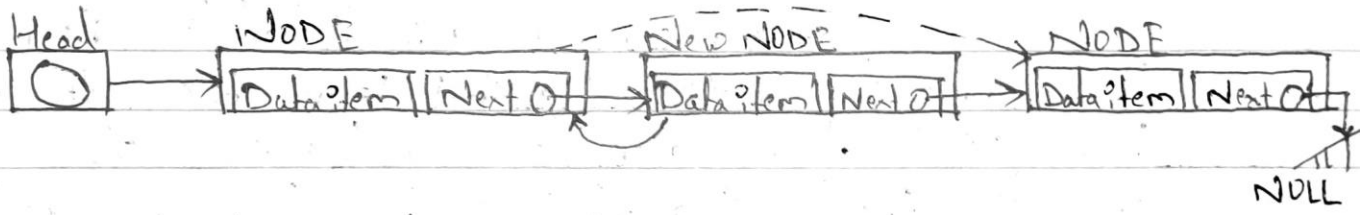
NewNode.next -> Right Node;

It should look like this;



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

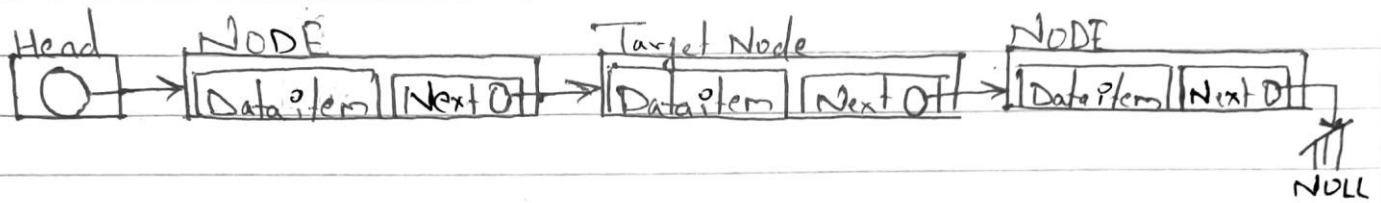
LeftNode.next -> NewNode;



Similar steps should be taken if the node is being inserted at beginning of list. While inserting it at the end, the second node last of list should point to new node & new node will put to end. NULL.

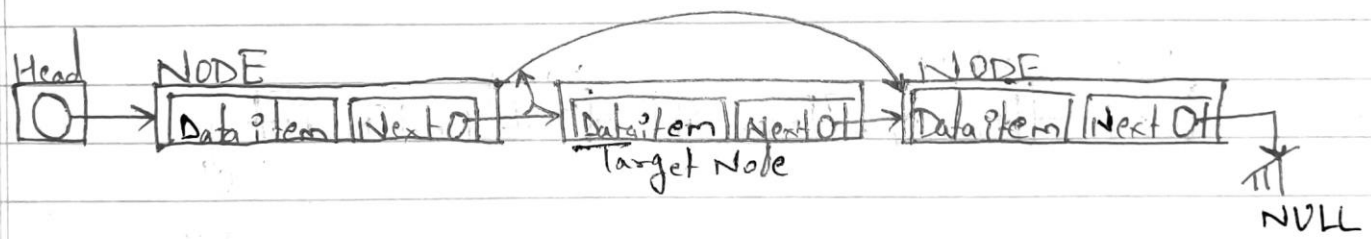
Deletion Operation:-

Deletion is also a more than one step process. We shall learn this pictorial representation. First, locate the process target to remove node, by using searching algorithms.



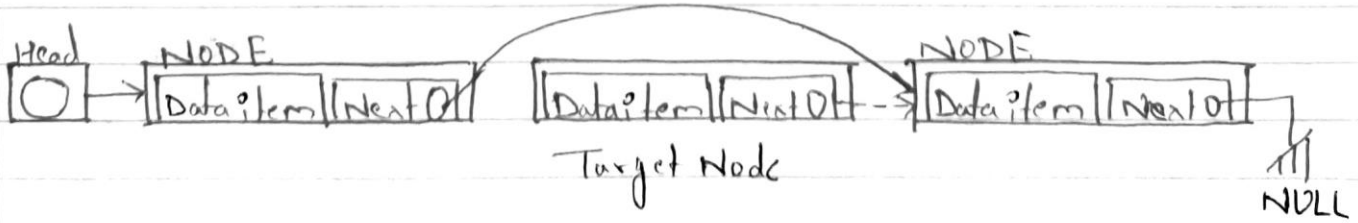
The previous node of target node now should point to next node of target node,

LeftNode.next -> TargetNode.next;

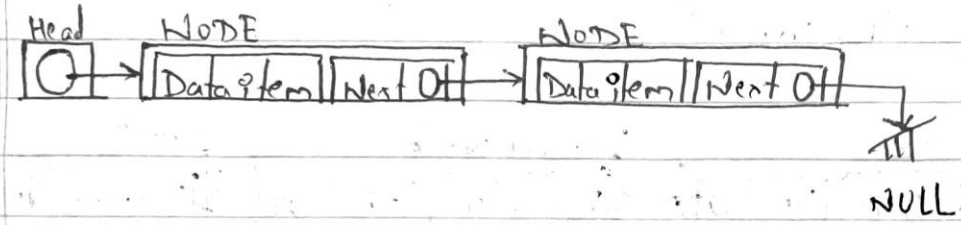


This will remove the link that was pointing to the target node. Now, using the following code, we will remove what target node is pointing at.

TargetNode.next -> NULL;

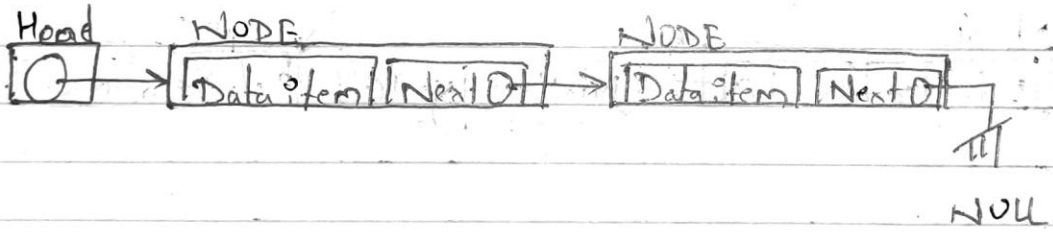


We need to use the deleted node. We can keep that in memory otherwise we can simply deallocate memory and wipeoff the target node completely.

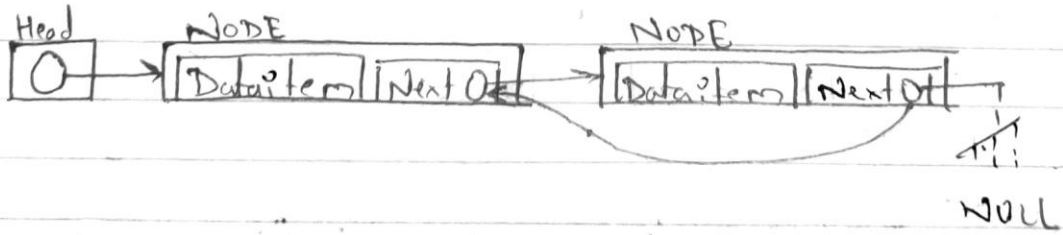


Reverse Operation:-

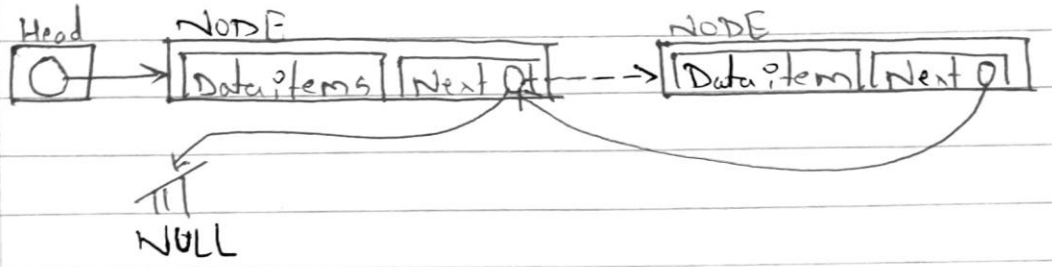
This operation is a through one. We need to make the last node to be pointed by head node & reverse the whole link list.



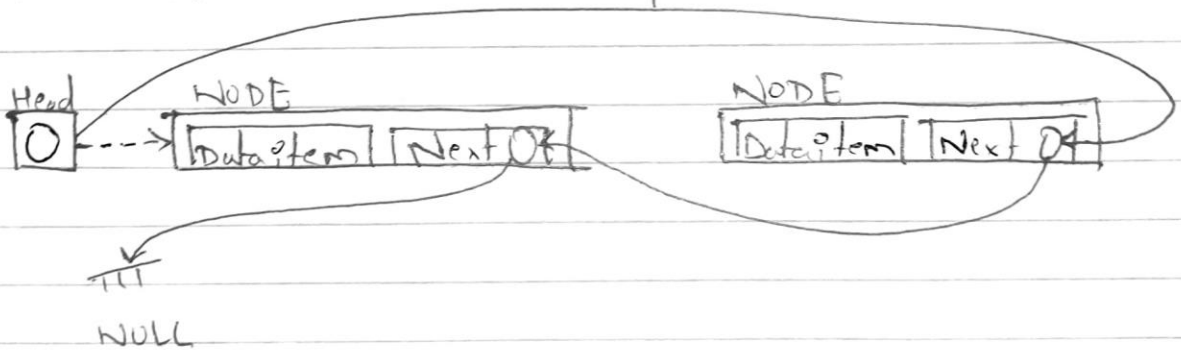
First we traverse to end of list. It should to pointing to null. Now, we shall make it point to previous node.



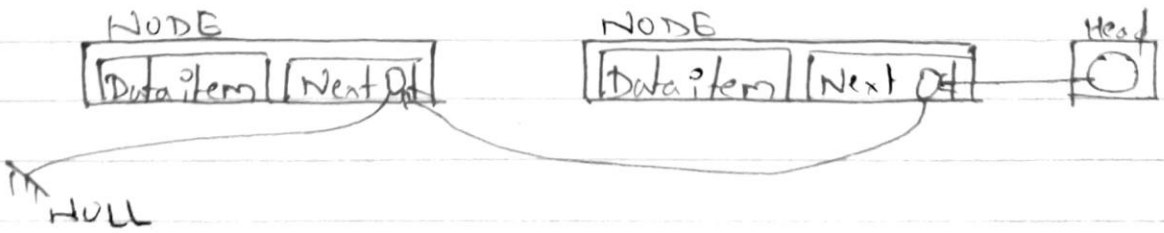
We have to make sure that the last node is the not lost node. So we'll have some temp node, which looks like head node pointing to last node. Now, we shall make all left side nodes point to their previous nodes one by one.



Except the node pointed by the head node, all node should point to their predessor, making them their new successor. The first node will point to NULL.



We will make the head node point to new first node by using temp node.



The linked list is now reversed.

Program Code:

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

struct node {
  string name;
  int MISno;
  node * next;
};
class member {
  public:
    node * header1, * header2;
  member() {
    header1 = NULL;
    header2 = NULL;
  }
  node * create();
  int count(node * x);
  void add(node * head);
  void del(node * head);
  void display(node * head);
  void rdisplay(node * cn);
  void concatinate();
  // void option();
};
int member::count(node * x) {
  node * cn = x;
  int count = 0;
  while (cn != NULL) {
    count++;
    cn = cn -> next;
  }
  return count;
}
node * member::create() {
  char ch;
  node * head;
  node * nn = new node;
  head = nn;
  cout << "Enter Name of president : ";
```

```
  cin >> nn -> name;
  cout << "Enter MIS no. of president : ";
  cin >> nn -> MISno;
  cout << endl;
  do {
    nn -> next = new node;
    nn = nn -> next;
    cout << "Enter Name of Member : ";
    cin >> nn -> name;
    cout << "Enter MIS no. of Member : ";
    cin >> nn -> MISno;
    cout << endl;
    cout << "Do you want to enter another member? (Y or y if yes) : ";
    cin >> ch;
    cout << endl;
  } while (ch == 'Y' || ch == 'y');
  nn -> next = new node;
  nn = nn -> next;
  cout << "Enter Name of secretary : ";
  cin >> nn -> name;
  cout << "Enter MIS no. of secretary : ";
  cin >> nn -> MISno;
  cout << endl;
  nn -> next = NULL;
  cout << "List is created! It has " << count(head) << " members!" <<
endl << endl;
  return head;
}
void member::display(node * head) {
  node * nn;
  nn = head;
  cout << "President : " << nn -> name << endl;
  cout << "MIS no. : " << nn -> MISno << endl << endl;
  nn = nn -> next;
  for (int i = 0; nn -> next != NULL; i++) {
    cout << "Member : " << nn -> name << endl;
    cout << "MIS no. : " << nn -> MISno << endl << endl;
    nn = nn -> next;
  }
  cout << "Secretary : " << nn -> name << endl;
  cout << "MIS no. : " << nn -> MISno << endl << endl;
  cout << "List has " << count(head) << " members!" << endl << endl;
```

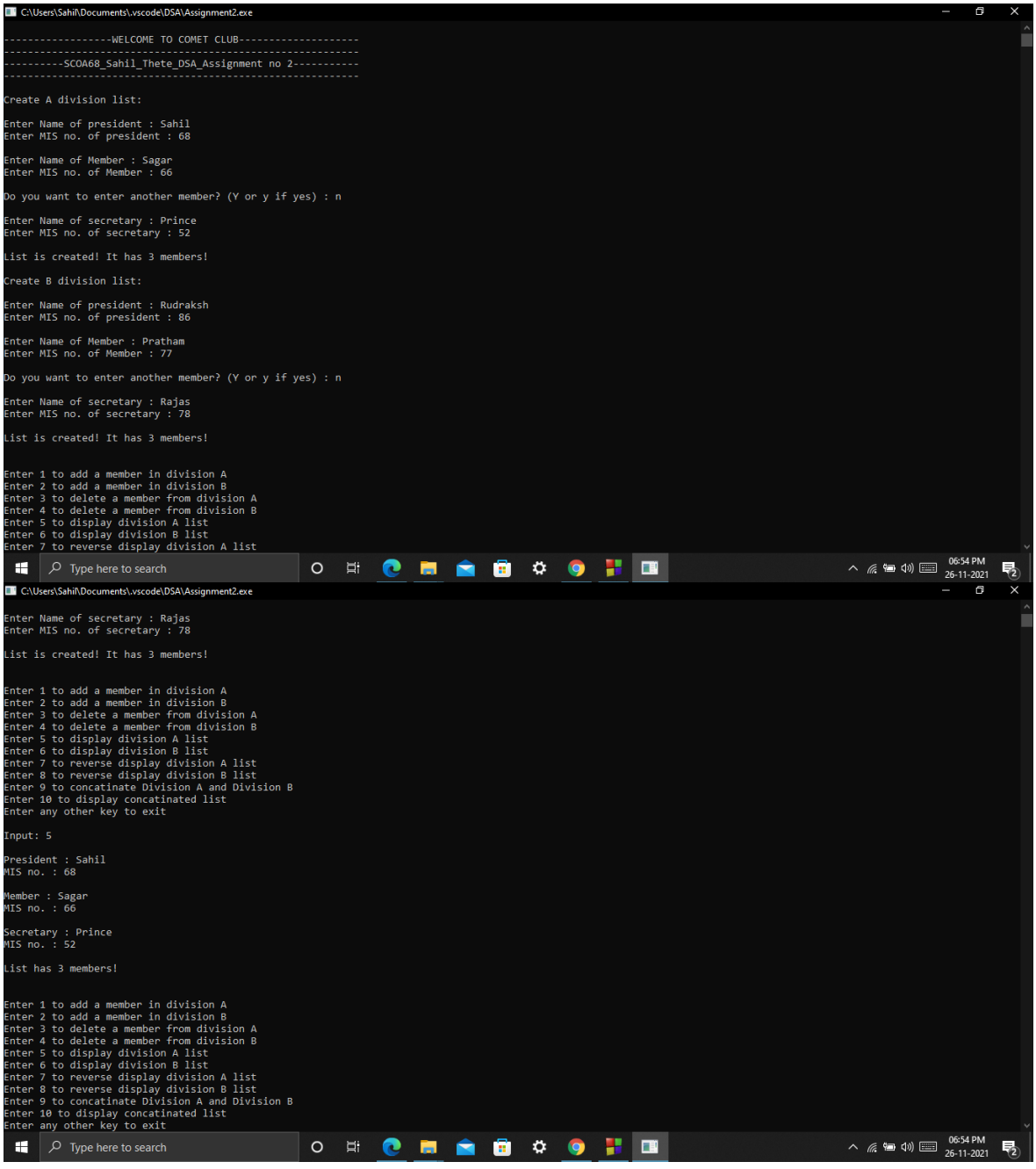
```
}
void member::rdisplay(node * cn) {
  if (cn == NULL)
    return;
  rdisplay(cn -> next);
  cout << cn -> name << endl;
  cout << cn -> MISno << endl;
  cout << endl;
}
void member::add(node * head) {
  int p;
  cout << "Enter the position where you want to add : ";
  cin >> p;
  node * nn, * temp;
  nn = head;
  node * an = new node;
  if (p == 1) {
    cout << "Enter Name of Member : ";
    cin >> an -> name;
    cout << "Enter MIS no. of Member : ";
    cin >> an -> MISno;
    cout << endl;
    an -> next = head;
    head = an;
    cout << "Member is added! List has " << count(head) <<
      " members now!\n\n";
  } else if (p == count(head) + 1) {
    for (int i = 0; nn -> next != NULL; i++) {
      nn = nn -> next;
    }
    cout << "Enter Name of Member : ";
    cin >> an -> name;
    cout << "Enter MIS no. of Member : ";
    cin >> an -> MISno;
    cout << endl;
    nn -> next = an;
    an -> next = NULL;
    cout << "Member is added! List has " << count(head) <<
      " members now!\n\n";;
  } else if (1 < p && p <= count(head)) {
    for (int i = 1; i < p; i++) {
      temp = nn;
```

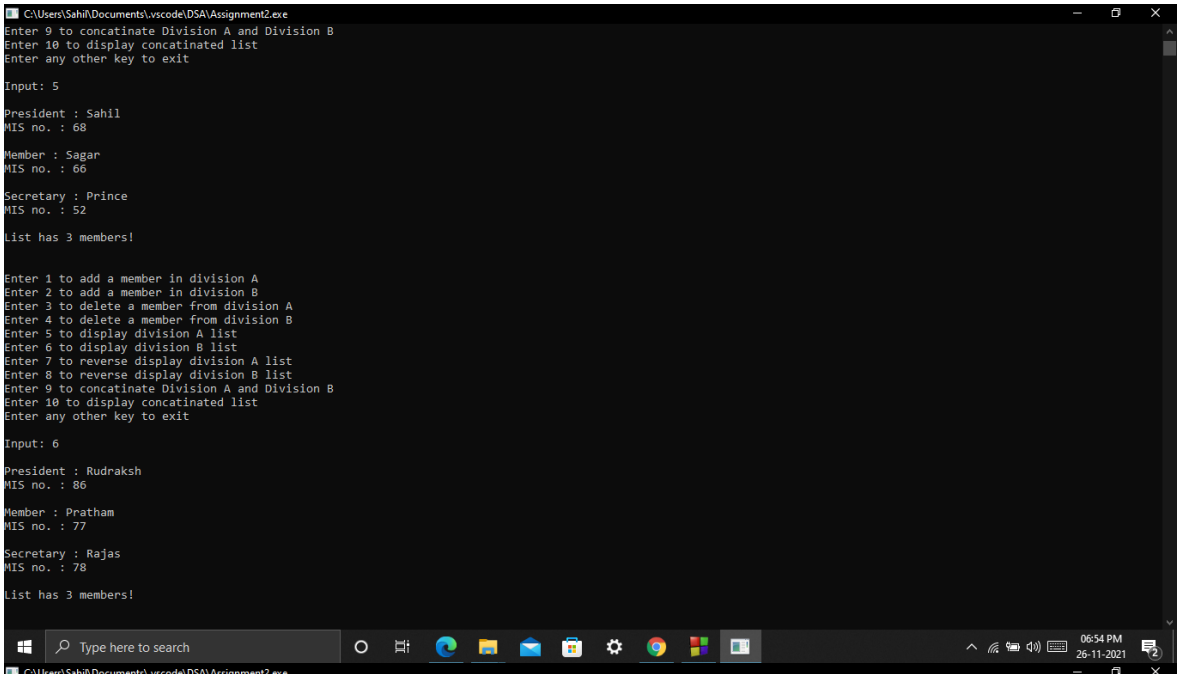
```
      nn = nn -> next;
    }
    cout << "Enter Name of Member : ";
    cin >> an -> name;
    cout << "Enter MIS no. of Member : ";
    cin >> an -> MISno;
    cout << endl;
    an -> next = nn;
    temp -> next = an;
    cout << "Member is added! List has " << count(head) <<
      " members now!\n\n";
  } else
    cout << "Invalid Position!\n\n";
}
void member::del(node * head) {
  int key;
  cout << "Enter the MIS no. of the student which is to be deleted : ";
  cin >> key;
  cout << endl;
  node * nn, * temp;
  nn = head;
  if (nn -> MISno == key) {
    head = nn -> next;
    delete(nn);
    cout << "Member deleted! List has " << count(head) <<
      " members now!\n\n";
  } else {
    while (nn -> MISno != key) {
      temp = nn;
      nn = nn -> next;
    }
    if (nn -> MISno == key && nn -> next == NULL) {
      temp -> next = NULL;
      delete(nn);
      cout << "Member deleted! List has " << count(head) <<
        " members now!\n\n";
    } else if (nn -> MISno == key) {
      temp -> next = nn -> next;
      delete(nn);
      cout << "Member deleted! List has " << count(head) <<
        " members now!\n\n";
    } else
```

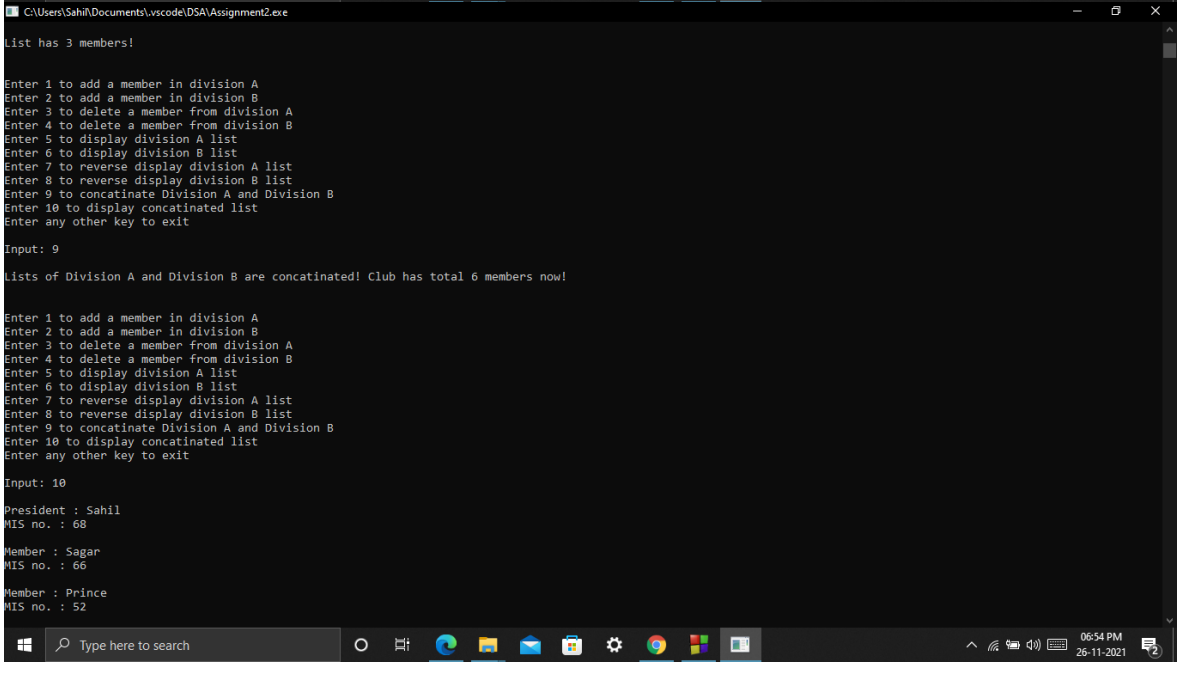
```
    cout << "Member not found!\n\n";
  }
}
void member::concatinate() {
  node * cn = header1;
  while (cn -> next != NULL)
    cn = cn -> next;
  cn -> next = header2;
  cout << "Lists of Division A and Division B are concatinated! Club has
total " <<
    count(header1) << " members now!\n\n";
}
int main() {
  int choice, g, a;
  member m;
  cout << "\n-----------------WELCOME TO COMET CLUB-----------------
---\n";
  cout << "------------------------------------------------------\n";
  cout << "----------SCOA68_Sahil_Thete_DSA_Assignment no 2----------
\n";
  cout << "------------------------------------------------------\n\n";
  cout << "Create A division list: " << endl << endl;
  m.header1 = m.create();
  cout << "Create B division list: " << endl << endl;
  m.header2 = m.create();
  while (true) {
    cout << endl << "Enter 1 to add a member in division A \nEnter 2 to
add a member in division B \nEnter 3 to delete a member from division A
\nEnter 4 to delete a member from division B \nEnter 5 to display
division A list \nEnter 6 to display division B list \nEnter 7 to reverse
display division A list \nEnter 8 to reverse display division B list \nEnter
9 to concatinate Division A and Division B \nEnter 10 to display
concatinated list \nEnter any other key to exit \n\nInput: ";
    cin >> choice;
    cout << endl;
    switch (choice) {
    case 1:
      m.add(m.header1);
      break;
    case 2:
      m.add(m.header2);
      break;
```

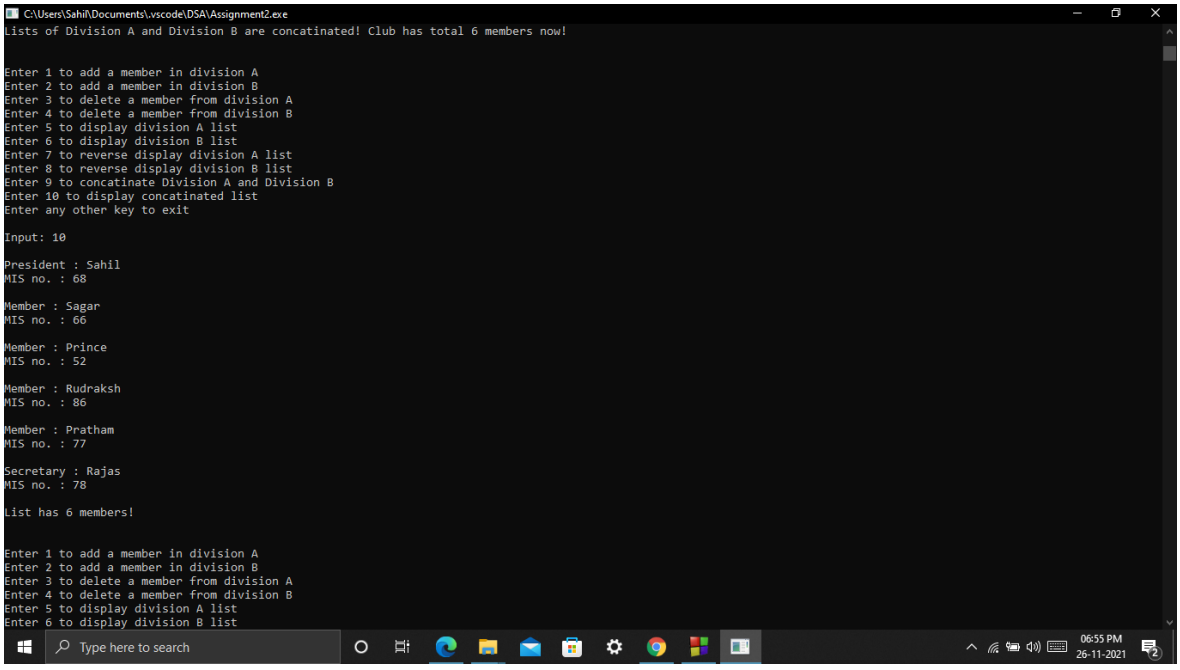
```
    case 3:
      m.del(m.header1);
      break;
    case 4:
      m.del(m.header2);
      break;
    case 5:
      m.display(m.header1);
      break;
    case 6:
      m.display(m.header2);
      break;
    case 7:
      m.rdisplay(m.header1);
      break;
    case 8:
      m.rdisplay(m.header2);
      break;
    case 9:
      m.concatinate();
      break;
    case 10:
      m.display(m.header1);
      break;
    default:
      return 0;
    }
  }
}
```

Program Output:




C:\Users\Sahil\Documents\.vscode\DSA\Assignment2.exe
Enter 9 to concatinate Division A and Division B
Enter 10 to display concatinated list
Enter any other key to exit
Input: 5
President : Sahil
MIS no. : 68
Member : Sagar
MIS no. : 66
Secretary : Prince
MIS no. : 52
List has 3 members!
Enter 1 to add a member in division A
Enter 2 to add a member in division B
Enter 3 to delete a member from division A
Enter 4 to delete a member from division B
Enter 5 to display division A list
Enter 6 to display division B list
Enter 7 to reverse display division A list
Enter 8 to reverse display division B list
Enter 9 to concatinate Division A and Division B
Enter 10 to display concatinated list
Enter any other key to exit
Input: 6
President : Rudraksh
MIS no. : 86
Member : Pratham
MIS no. : 77
Secretary : Rajas
MIS no. : 78
List has 3 members!
Type here to search
06:54 PM
26-11-2021


C:\Users\Sahil\Documents\.vscode\DSA\Assignment2.exe
List has 3 members!
Enter 1 to add a member in division A
Enter 2 to add a member in division B
Enter 3 to delete a member from division A
Enter 4 to delete a member from division B
Enter 5 to display division A list
Enter 6 to display division B list
Enter 7 to reverse display division A list
Enter 8 to reverse display division B list
Enter 9 to concatinate Division A and Division B
Enter 10 to display concatinated list
Enter any other key to exit
Input: 9
Lists of Division A and Division B are concatinated! Club has total 6 members now!
Enter 1 to add a member in division A
Enter 2 to add a member in division B
Enter 3 to delete a member from division A
Enter 4 to delete a member from division B
Enter 5 to display division A list
Enter 6 to display division B list
Enter 7 to reverse display division A list
Enter 8 to reverse display division B list
Enter 9 to concatinate Division A and Division B
Enter 10 to display concatinated list
Enter any other key to exit
Input: 10
President : Sahil
MIS no. : 68
Member : Sagar
MIS no. : 66
Member : Prince
MIS no. : 52
Type here to search
06:54 PM
26-11-2021


C:\Users\Sahil\Documents\.vscode\DSA\Assignment2.exe
Lists of Division A and Division B are concatinated! Club has total 6 members now!
Enter 1 to add a member in division A
Enter 2 to add a member in division B
Enter 3 to delete a member from division A
Enter 4 to delete a member from division B
Enter 5 to display division A list
Enter 6 to display division B list
Enter 7 to reverse display division A list
Enter 8 to reverse display division B list
Enter 9 to concatinate Division A and Division B
Enter 10 to display concatinated list
Enter any other key to exit
Input: 10
President : Sahil
MIS no. : 68
Member : Sagar
MIS no. : 66
Member : Prince
MIS no. : 52
Member : Rudraksh
MIS no. : 86
Member : Pratham
MIS no. : 77
Secretary : Rajas
MIS no. : 78
List has 6 members!
Enter 1 to add a member in division A
Enter 2 to add a member in division B
Enter 3 to delete a member from division A
Enter 4 to delete a member from division B
Enter 5 to display division A list
Enter 6 to display division B list
Type here to search
06:55 PM
26-11-2021

Conclusion:

This we we implemented, operations on singly linked list.