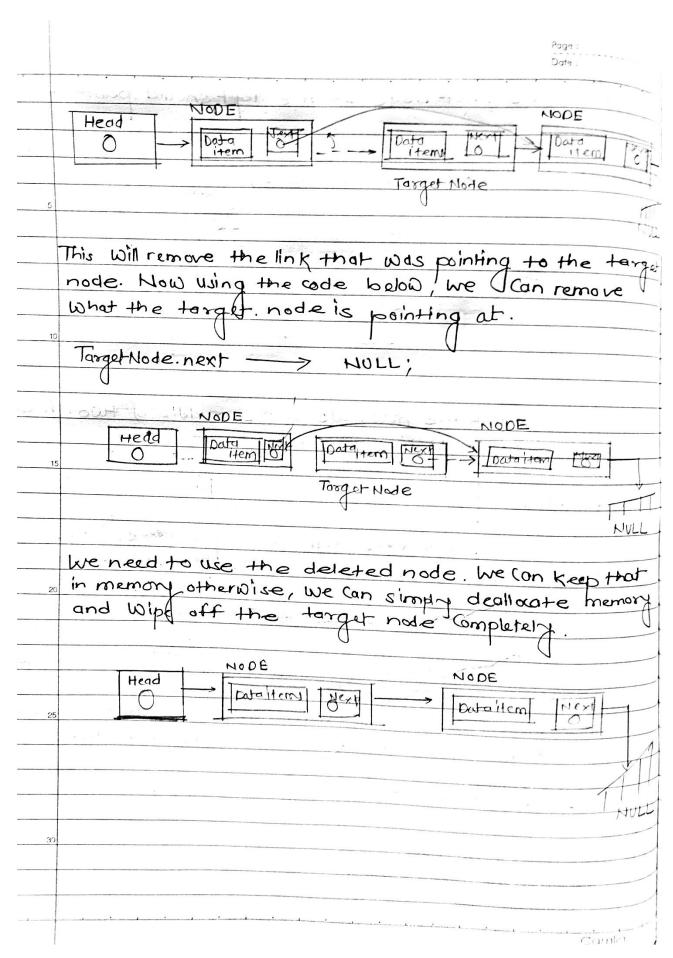
DSA Experiment No.-02

	Experiment No. (02) Page: Date:
AIM	named 'COMET' students of second, third and find
	year of department can be granted membership of club and last node is reserved for the secretary of
	club.
	Write a program to main club member's information using singly linked list. Store student MIS Regestration NO. I and Name, Write functions to add
	(a) Add and delete the members as well as prejides!
10	or even secretary
	(b) Compute total No of members of club
	(c) Display members
	(d) Display list in reverse order using recursion
15	(e) Two linked lists exists for two divisions. Concatenate too lists.
	. WO 113 3
Theory	7 com
U	A linked list is the sequence of data structure which are
	Connected together via links. Linked list Is a sequence of links
20	which Contains items. Each link Contains a Connection to another
	link.
	Linked list is the second most used data structure
	after array. Following are the important terms to under- stand the Concept of Linked list:
	Starra the Sheep of Linked list.
25	• 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 nextlink called Next.
· vi	They want to the
3,	
The second secon	Milror Company

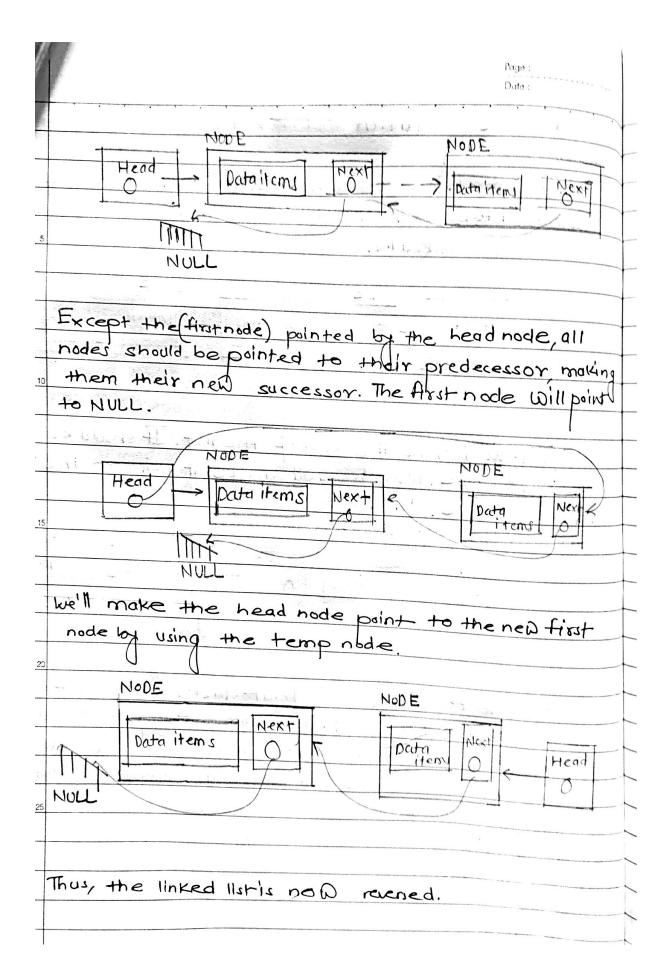
Linked list — A linked list Contains the connection line to the first link Called first. Linked list Copresentation Linked list Combe visualized as a chain of nodes, linked list Combe visualized as a chain of nodes, linked surry node points to the next node. NODE NODE NODE NODE NODE NODE Head NODE Head NODE Head NODE Head NODE Head NODE NODE Head NODE Head NODE Head NODE Head NODE NODE Head NODE Head NODE Head NODE NODE Head NODE Head NODE NODE Head NODE NODE NODE Head NODE NODE NODE Head NODE NO						Page : Date :		
Linked list Con be visualized as a choin of nodes, who every node points to the next node. NODE NODE	Linked 1	121	A linked	1 115+	Contains	the conne	chian	link
Linked list Can be visualized as a chain of nodes, who every node points to the next node. NODE NODE	to the	first link	Called fi	77+.				
Linked list Colo be visualized as a chain of nodes, who every node points to the next node. NODE NOD			. 1	1/10/11	1 celas	21 /37 m		
As a part of above illustration, following are the important points considered, to be. Linkedist Contains a link exement Colled first. Each link amies a data fields and a link field Caller next Each link is linked with it's next using the next link. Lask link Graies a link as null many the end of the to Types of the linked lish Following are the vorious types of the linkedlish Following are the vorious types of the linkedlish								
NoDE Head Down them May Described from May NO As a past of above illustration, following are the important points Considered, to be. Linkedist Contains a link element Colled first. Each link amies a data fields and a link field Called next Each link is linked with it's next using the next link. Lask link Carries a link as null many the end of the to the linked lish Types of the linked lish Following are the vorious types of the linked lish Following are the vorious types of the linked lish						ain of nod	es, W	hen
As a past of above illustration following are the important points considered to be. Linkedist Contains a link exement Colled first. Each link wies a data fields and a link field Called next Fach link is linked with it's next using the next link. Lask link Carries a link as null many the end of the to the linked lish Types of the linked lish Following are the various types of the linked lish	every no	ide points						
Head Detailed Detailed Detailed Detailed Detailed		None					,	
As a past of above illustration, following are the important points considered to be. Linkedlist Contains a link element Called first. Each link amies a data fields and a link field Called next Fach link is linked with it's next using the next link. Lask link Corries a link as null mark the end of the to Types of the linked lish Following are the vorious types of the linked lish	Head	-		10000				
points considered to be. to be. Linkedist Contains allow exement Colled First. Each link Comies a data fields and a link field Called next Fach link is linked with it's next using the next link. Lask link Comies a link as null many the end of the to Types of the linked list Following are the various types of the linked list Following are the various types of the linked list	0-	Dain It a	m 1831	> Dute	item - The	X Data	Her [0
points considered to be. to be. Linkedist Contains a link element Colled first. Each link Comies a data fields and a link field Called next Teach link is linked with it's next using the next link. Lask link Carries a link as null mark the end of the to Types of the linked list Following are the various types of the linked list Following are the various types of the linked list						Malinana		
points considered to be. to be. Linkedist Contains a link element Colled first. Each link Comies a data fields and a link field Called next Each link is linked with it's next using the next link. Lask link Carries a link as null mark the end of the to Types of the linked lish Following are the vorious types of the linked lish Following are the vorious types of the linked lish						4		7
points considered to be. to be. Linkedist Contains a link element Colled first. Each link Comies a data fields and a link field Called next Each link is linked with it's next using the next link. Lask link Carries a link as null mark the end of the to Types of the linked lish Following are the vorious types of the linked lish Following are the vorious types of the linked lish				-	-			MI
points considered to be. to be. Linkedist Contains a link element Colled first. Each link Comies a data fields and a link field Called next Teach link is linked with it's next using the next link. Lask link Carries a link as null mark the end of the to Types of the linked list Following are the various types of the linked list Following are the various types of the linked list			- 142					NUL
Linkedist Contains a link element Colled First. Each link Comies a data fields and a link field Called next Teach link is linked with it's next using the next link. Lask link Carries a link as null mark the end of the to Types of the linked lish Following are the various types of the linked lish Following are the various types of the linked lish					lowing a	are the in	npor	tan
Linkedist Contains allow exement Colled First. - Each link Comies a data fields and a link field Called next - Each link is linked with it's next using the next link. - Lask link Carries a link as null mark the end of the to Types of the linked lish Following are the various types of the linked lish Following are the various types of the linked lish	4		1, to be		U		1	
- Each link Comies a data fields and a link field Called next - Each link is linked with it's next using the next link. - Lask link Carries a link as null mark the end of the to Types of the linked lish Following are the various types of the linked lish Following are the various types of the linked lish			· · · · · · · · · · · · · · · · · · ·					
next - Each link is linked with it's next using the next link. - Lask link Carries a link as null many the end of the to Types of the linked list Following are the various types of the linked list - Following are the various types of the linked list	Fach	late Conta	م م ما ا	الر ع	ement	alled Ho	13r.	
Each link is linked with it's next using the next link. Lask link Carries a link as null many the end of the to Types of the linked list Following are the various types of the linked list.	nexh	ink one	<u> </u>	ta trei	as ana.	a link the	la a	ill ed
Lask link Carries a link as null mark the end of the to Types of the linked lish Following are the various types of the linked lish ?	-	ak ic linkes	with it's	nex.	+ usina	the next	t lin	k
Types of the linked list >> Following are the various types of the linked list >>					/			
Following are the vorious types of the linkedlish ->					10			
Following are the vorious types of the linkedlish ->	Types of	the linke	d list	>				
9	Following a	ie the vo	rious typ	oes of	the link	red lish —	->	
Simple linkedlish Them novigation is forward only. - Doubly linkedlish Thems Can be navigated forward and backward.	7							
- Roubly linkedist - Items Can be navigated forward and backward.	Simple	11n ked list		Item r	navigation	is forwar	rd on t	¥.
backward.	- Quby li	n kedlist —	>Items (îan be	naviga	ted forwar	rd an	4)
	Jb	ackword.			7			
		`						
Tr.								
	ksh.karne	.cs@ghrce	m.raison	i net				

	Page: Date:
+	Circular linked list Last item Contains link of the
	Circular linked list Last item Contains link of the first element as next and the first element has
	link to the last element as previous.
	in the second of
5	Basic Operation.
	CONTRACTOR OF STREET
1	Adding new node to linked list is more than one step ac
	And location where it has to be inserted.
10	
	Nobe
	Head Data items Next Data items New
	Lampara and Lambara and Lambar
15	Data Items Next
	NEW NAME
	Imagine that we are inverting a node B (New node) both
1	Imagine that we are inserting a node B (New node) both Acceptnode) and C (Rightnode). Then point B. next too
1	
20	NewHodenext -> Righthlode;
-	It should look like this
1	NoDE
1	A constant of the constant of
5	Destritery Next
1	
1	
1	Data items MSY
0	*** The state of t
	NEW MODE

This will but the new node in the middle of two. The new list should look like this Deletion operation Deletion is also a more than one step process. We shall learn with pictorial representation. First locate the thought to be removed by using searching algorithm to Deletion.	new node	e next node at d	the left should	point 75 m
This will but the new node in the middle of two. The new list should look like this Deletion a peration Deletion is also a more than one step process. We shall be to be removed by using searching algorithm with pictorial representation. First locate the thought head head head head head head head head	1120			
This will but the new node in the middle of two. The new list should look like this Deletion a peration Deletion is also a more than one step process. We shall be to be removed by using searching algorithm with pictorial representation. First locate the thought head head head head head head head head	leftNode	next -> NeDI	vode:	- 122
This will but the new node in the middle of two. The new list should look like this Deletion operation Deletion is also a more than one step process. We shall learn with pictorial representation. First locate the thought node to be removed by using searching algorithms to Deletion. The left (previous) node of the target node now show point to the next node of the target node.		4		
This will but the new node in the middle of two. The new list should look like this Deletion operation Deletion is also a more than one step process. We shall be as not be removed by using searching algorithm to the removed by using searching algorithm west path west path litems The left (previous) node of the target node now show point to the next node of the target node.	Big Discourse in which we like the apparent of the control of	WOO F		anticolor del reconsidera establisha establi
This will but the new node in the middle of two. The new list should look like this Deletion operation Deletion is also a more than one step process. We shall learn with pictorial representation. First locate the thought node to be removed by using searching algorithms to DE Head The left (previous) node of the target node now show point to the next node of the target node.	Head	loate Heart Next	1 Kym - 1 200 - 1	note Next
This will but the new node in the middle of two. The new list should look like this Deletion operation Deletion is also a more than one step process. We shall be an with pictorial representation. First locate that node to be removed by using searching algorithms THE LEFT (previous) node of the target node now show point to the next node of the target node.		The Administrative and the second		1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
This will but the new node in the middle of two. The new list should look like this Deletion operation Deletion is also a more than one step process. We shall be an with pictorial representation. First locate that node to be removed by using searching algorithms THE LEFT (previous) node of the target node now show point to the next node of the target node.			- 11 12 / Marie 1	3.5 Carter
This will but the new node in the middle of two. The new list should look like this Deletion operation Deletion is also a more than one step process. We shall be as with pictorial representation. First locate the thought node to be removed, by using searching algorithms head Head Deletion is also a more than one step process. We shall be as a searching algorithm of the next node of the target node now show point to the next node of the target node.		1900	Account to the second s	
Deletion operation Deletion is also a more than one step process. We shall be as with pictorial representation. First locate that node to be removed by using searching algorithms Head Data Next Data Next Data Next Data Next New Head Next Next Next Next Next Next Next Next			Manager of the second of the s	2 1 2 - 11 - E
Deletion operation Deletion is also a more than one step process. We shall be as with pictorial representation. First locate that node to be removed by using searching algorithms Head Data Next Data Next Data Next Data Next New Head Next Next Next Next Next Next Next Next				
Deletion operation Deletion is also a more than one step process. We shall be as with pictorial representation. First locate that node to be removed by using searching algorithms Head Data Next Data Next Data Next Data Next New Head Next Next Next Next Next Next Next Next	This will	but the new node	in the middle	of two. The
Deletion operation Deletion is also a more than one step process. We shall be as with pictorial representation. First locate the three node to be removed, by using searching algorithms with the part went part went part went part went part went them. The left (previous) node of the target node now show point to the next node of the target node.				
Deletion is also a more than one step process. We shall be as with pictorial representation. First locate the three node to be removed by using searching algorithms with the part west with them thems of the start of the next node of the target node now show point to the next node of the target node.	. 1000		And the second s	
Deletion is also a more than one step process. We shall be as with pictorial representation. First locate the three node to be removed by using searching algorithms with the part west with them thems of the start of the next node of the target node now show point to the next node of the target node.	Deletion	Operation	* ;	
Head para Next Data Next Data Next The left (previous) node of the target node now show point to the next node of the target node.	20121011			
Head para Next Data Next Data Next The left (previous) node of the target node now show point to the next node of the target node.	Deletion i	s also a more than	none step proc	ess. We sho
Head para Next Data Next Data Next Data Next Next Data Next Next Next Data Next Next Next Next Next Next Next Next	leam wit	h pictorial represe	ntation. First	locate thete
The left (previous) node of the target node now show point to the next node of the target node.	node to	be removed by	using searchi	ng algorithm.
The left (previous) node of the target node now show point to the next node of the target node.	+	KIODE	HODE	
The left (previous) node of the target node now show point to the next node of the target node.	Head	1 11. 1 1 2 1 1		
The left (previous) node of the target node now show point to the next node of the target node.				Trens C
point to the next node of the target node.			10,000	NIVLL
point to the next node of the target node.	The left	· (previous) node of	the target no	ide now shou
	point t	o the next node	of the target	- node.
Lefthode next -> Targethode next;				
	Leftho	denext -> Tan	getNode next;	
			/	
· ·				



					Page : Date :	********
Revers	e Op	eration -		······································	, , , , , , , , , , , , , , , , , , , ,	
	agent and the second second					per la completa de la completa del la completa de la completa del la completa de la completa del la completa de la completa de la completa del la completa de la completa de la completa del la co
This op	eration	is through	one. W	e need t	o make t	he las
node -	tobe:	pointed by	the he	ad node	and rev	erse
the WI				1 1/2 1/2 1		
<u> </u>		NODE	Annual Control of the	NODE	+	
~	Head	Data item	1 O N(XI	-> Data	items Ne	*+
	, -;- x	too with				\Box
						NU
Fixet W	re trove	erse the end	d of the	-list. It	should be	e pair
na to 1	NULL.	NOW, we sh	all mak	e it poin	nt to the	e pre
lous	node.	The Action of		and a second of the	~ ~ * * * *	•
-		Node	The state of the	Node		-
F	Head O	Data items	next	Datai	teny Next	
		1 . 4		2 7 4	7	TII
we hav	e to r	make sure	that	last nod	e is not	+hc
	~	Ca 111h	mire Som	e tem	p noae,	Whid
10014	ike H	e head no	de pal	nting to	the la	-nog
NOW,	we st	all make c	ill left.	on Usbiz	des pool	W TC
their	previo	ne head no hall make co ous nodes	one by	one.		
			U			
	1					



Program code:

```
#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;
  }
rudraksh.karpe.cs@ghrcem.raisoni.net
```

```
}
void list::create()
{
  int no;
  string nam;
  if (start == NULL)
  {
    cout << "Enter MIS number: ";</pre>
    cin >> no;
    cout << "Enter name: ";</pre>
    cin >> nam;
    cout << nam;</pre>
    start = new node(no, nam);
    cout << "\n===== List Created =====";
  }
  else
  {
    cout << "\nList is already created.";</pre>
  }
void list::display()
```

```
{
  node *t;
  t = start;
  if (start == NULL)
    cout << "\nList is Empty";</pre>
  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: ";</pre>
    cin >> no;
    cout << "Enter name: ";</pre>
    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;
rudraksh.karpe.cs@ghrcem.raisoni.net
```

```
string nam;
  node *t;
  if (start == NULL)
    create();
  else
  {
    cout << "\nEnter MIS number: ";</pre>
    cin >> no;
    cout << "Enter name: ";</pre>
    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:";</pre>
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: ";</pre>
  cin >> no;
  cout << "Enter name: ";</pre>
```

```
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..";</pre>
  else
    t = start;
    start = start->next;
```

```
t->next = NULL; //Not necessary
    delete t;
    cout << "\nPresident deleted..";</pre>
  }
}
void list::deleteByValue()
{
  int no, flag = 0;
  node *t, *prev;
  if (start == NULL)
    cout << "\nList/Club is empty;";</pre>
  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)
rudraksh.karpe.cs@ghrcem.raisoni.net
```

```
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</pre>
secretary cannot be deleted.";
  }
void list::deleteAtEnd()
{
  node *t, *prev;
rudraksh.karpe.cs@ghrcem.raisoni.net
```

```
t = start;
  if (start == NULL)
    cout << "\nClub is Empty..";</pre>
  else
  {
    while (t->next != NULL)
    {
       prev = t;
       t = t->next;
    }
     prev->next = NULL;
    delete t;
    cout << "\nSecretary Deleted.";</pre>
  }
int list::computeTotal()
{
  node *t;
  int count = 0;
  t = start;
  if (start == NULL)
```

```
{
    cout << "\nList is empty.";</pre>
    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.";</pre>
  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.";</pre>
display();
```

```
}
void list::concatList(list &q1)
{
  node *t, *p;
  t = q1.start;
  if (t == NULL)
  {
    cout << "\nList 2 is empty";</pre>
    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";</pre>
  display();
int main()
```

```
{
  list *l;
  int choice, selectList;
  list 11, 12;
  | = &|1;
X:
  cout << "Welcome to COMET GHRCEM!" << endl;</pre>
  cout << "SCOB86_Rudraskh Karpe" << endl;</pre>
  cout << "\nSelect List\n1.List 1(Div-1)\n2.List 2(Div-</pre>
2)\nEnter choice: ";
  cin >> selectList;
  if (selectList == 1)
  {
    | = & |1;
  else if (selectList == 2)
  {
    I = &I2;
  }
  else
rudraksh.karpe.cs@ghrcem.raisoni.net
```

```
cout << "\nWrong list Number.";</pre>
    goto X;
  }
  do
  {
    cout << "\n1. create\n2.Insert President\n3.Insert</pre>
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 List ++--##"
       << "\n12.Combine lists\n13.Reverse Display\n0.</pre>
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 &</pre>
Secretary): " << I->computeTotal();
       break;
rudraksh.karpe.cs@ghrcem.raisoni.net
```

```
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";</pre>
    }
  } while (choice != 0);
  cout << "\n====== GOOD BYE
=======\n";
  return 0;
```

Output:

Interface of the program

```
PS R:\GHRCEM\DSA Lab> cd "r:\GHRCEM\DSA Lab\" ; if ($?) { g++ main_new.cpp -o main_new } ; if ($?) { .\main_new }
Welcome to COMET GHRCEM!
SCOB86_Rudraskh Karpe
Select List
1.List 1(Div-1)
2.List 2(Div-2)
Enter choice: 1
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
```

Adding a member

```
PS R:\GHRCEM\DSA Lab> cd "r:\GHRCEM\DSA Lab\" ; if ($?) { g++ main_new.cpp -o main_new } ; if ($?) { .\main_new }
Welcome to COMET GHRCEM!
SCOB86_Rudraskh Karpe
Select List
1.List 1(Div-1)
2.List 2(Div-2)
Enter choice: 1
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 MIS number: 12345
Enter name: Rohan
===== List Created =====
```

Adding President

```
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:
                        2
Enter MIS number: 19105418
Enter name: Rudraksh
Inserted Rudraksh at the beginning.
```

Adding Secretary

```
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
Exit
ENter your choice:
Enter MIS number: 984278
Enter name: Mihir
```

Display List

```
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
                    5
ENter your choice:
===== List: =====
19105418 Rudraksh
12345 Rohan
984278 Mihir
```

Inserting after position member

```
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 MIS No. after do you want insert:19105418
Enter MIS number: 78
Enter name: Mohit
```

```
ENter your choice: 5

===== List: =====
19105418 Rudraksh
78 Mohit
12345 Rohan
984278 Mihir
```

Reversing the List

```
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: 13
MIS NO:984278 Name: Mihir
MIS NO:12345 Name: Rohan
MIS NO:78 Name: Mohit
MIS NO:19105418 Name: Rudraksh
```

Switching between the List 1 and List 2

```
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:
Welcome to COMET GHRCEM!
SCOB86_Rudraskh Karpe
Select List
1.List 1(Div-1)
2.List 2(Div-2)
Enter choice: 2
```

Adding Member

```
ENter your choice: 1
Enter MIS number: 895385
Enter name: Gautam
Gautam
====== List Created =====
```

Adding President

```
Enter your choice: 2

Enter MIS number: 53893

Enter name: Dinesh
Inserted Dinesh at the beginning.
```

ENter your choice: 3

Enter MIS number: 35837586

Enter name: Kunal

Displaying List 2

ENter your choice: 5

===== List: =====

53893 Dinesh

895385 Gautam

35837586 Kunal

Concatenating List 1 and List 2

```
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: 12
After concatenationlist
===== List: =====
19105418 Rudraksh
78 Mohit
12345 Rohan
984278 Mihir
53893 Dinesh
895385 Gautam
35837586 Kunal
```

Deleting the President

```
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: 6
President deleted..
```

Deleting Secretary

```
President deleted..
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
Exit
ENter your choice:
Secretary Deleted.
```

List after Deleting Members

```
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
Exit
ENter your choice:
===== List: =====
12345 Rohan
984278 Mihir
53893 Dinesh
895385 Gautam
```