Name- Abhinav Kumar
PRN- 21070126006
Branch- AIML-A1
Lab Assignment- Singly LinkList

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

// Creating structure for node declaration
struct node
{
	int data;
	struct node* link;
};

// Traverse the list
void traverse(struct node* head)
{
	struct node* temp = head;
	cout << endl;
	while (temp != NULL)
	{
		//cout << "\nAddress: " << temp << endl;
		//cout << "Value: " << temp->data << endl << endl;
		cout << temp->data << " -> ";
		temp = temp->link;
	}
}

// Insert at the beginning of the list
struct node* insert_beg(struct node* head)
{
    int n;
    cout << "Enter data of new node: ";
    cin >> n;
    struct node* newnode = NULL;
    newnode = (struct node*)malloc(sizeof(struct node*));
    newnode->data = n;
    newnode->link = NULL;
    newnode->link = head;
    head=newnode;
    return head;
}

// Insert at a position in the list
void pos_insert(struct node* head)
{
	int n, position;
	cout << "Enter data of new node: ";
	cin >> n;
```

```
        cout << "Enter position to enter: ";
        cin >> position;
        struct node* temp = head;
        struct node* newnode = NULL;
        for (int i = 1; i < position - 1; i++)
        {
                temp = temp->link;
        }
        newnode = (struct node*)malloc(sizeof(struct node*));
        newnode->data = n;
        newnode->link = NULL;
        newnode->link = temp->link;
        temp->link = newnode;
}

// Insert at the end of the list
void insert_end(struct node* head)
{
    int n;
    cout << "Enter data of new node: ";
    cin >> n;
    struct node* newnode = NULL;
    newnode = (struct node*)malloc(sizeof(struct node*));
    newnode->data = n;
    newnode->link = NULL;
    struct node* temp = head;
    while (temp->link != NULL)
    {
        temp = temp->link;
    }
    temp->link = newnode;
}

//Delete from the beginning of the list
struct node* delete_beg(struct node* head)
{
        struct node* temp = head;
        head = head->link;
        free(temp);
        return head;
}

//Delete from end of the list
void delete_end(struct node* head)
{
        struct node* temp = head;
        while (temp->link->link != NULL)
        {
                temp = temp->link;
        }
        temp->link = NULL;
        free(temp->link);
}

//Delete from a position in the list
void pos_delete(struct node* head)
{
        int position;
        cout << "Enter position to delete: ";
        cin >> position;
        struct node* temp = head;
        struct node* temp2 = temp;
        for (int i = 1; i < position - 1; i++)
        {
                temp = temp->link;
        }
        temp2 = temp->link;
        temp->link = temp2->link;
        free(temp2);
}

//Concatenate two lists
void concat(struct node* head)
{
        struct node* head2 = NULL;
        struct node* second2 = NULL;
        struct node* third2 = NULL;

        head2 = (struct node*) malloc(sizeof(struct node));
        second2 = (struct node*)malloc(sizeof(struct node));
        third2 = (struct node*)malloc(sizeof(struct node));

        head2->data = 102;
        head2->link = second2;
        second2->data = 269;
        second2->link = third2;
        third2->data = 420;
        third2->link = NULL;

        struct node* temp = head;
        while (temp->link != NULL)
        {
                temp = temp->link;
        }
        temp->link = head2;
}

//Reverse the list
void reverse(struct node* head)
{
	struct node* temp = head;
	struct node* next = NULL;
	struct node* prev = NULL;
	while (temp != NULL)
	{
		next = temp->link;
		temp->link = prev;
		prev = temp;
		temp = next;
	}
	head = prev;
}

//Search an element in the list
void search(struct node* head)
{
	int n;
	cout<<"Enter element to search: ";
	cin>>n;
	struct node* temp = head;
	int pos=1;
	while(temp != NULL)
	{
		if(temp->data == n)
		{
			cout<<"\nElement found at position: "<<pos<<endl;
			break;
		}
		temp = temp->link;
		pos++;
	}
	if(temp == NULL)
	{
		cout<<"\nElement does not exist!"<<endl;
	}
}

//Menu of the program
void menu(struct node* head)
{
	int choice;

	cout<<"\n\n1. Traverse the list"<<endl;
	cout<<"2. Insert at the beginning of the list"<<endl;
	cout<<"3. Insert at a position in the list"<<endl;
	cout<<"4. Insert at the end of the list"<<endl;
	cout<<"5. Delete from the beginning of the list"<<endl;
	cout<<"6. Delete from the end of the list"<<endl;
	cout<<"7. Delete from a position in the list"<<endl;
	cout<<"8. Concatenate two lists"<<endl;
```

```
cout<<"9. Reverse a list"<<endl;
cout<<"10. Search an element in list"<<endl;
cout<<"11. Exit"<<endl;
cout<<"Enter choice: ";
cin>>choice;

if (choice==1)
{
        traverse(head);
}
else if (choice==2)
{
        head = insert_beg(head);
        traverse(head);
}
else if (choice==3)
{
        pos_insert(head);
        traverse(head);
}
else if (choice==4)
{
        insert_end(head);
        traverse(head);
}
else if (choice==5)
{
        head = delete_beg(head);
        traverse(head);
}
else if (choice==6)
{
        delete_end(head);
        traverse(head);
}
else if (choice==7)
{
        pos_delete(head);
        traverse(head);
}
else if (choice==8)
{
        concat(head);
        traverse(head);
}
else if (choice==9)
{
        reverse(head);
        traverse(head);
}
```

```
	else if(choice==10)
	{
		search(head);
	}
	else if (choice==11)
	{
		exit(0);
	}
	else
	{
		cout<<"\nInvalid choice"<<endl;
	}
	menu(head);
}

//main function
int main()
{
	struct node* head = NULL;
	struct node* second = NULL;
	struct node* third = NULL;

	head = (struct node*) malloc(sizeof(struct node));
	second = (struct node*)malloc(sizeof(struct node));
	third = (struct node*)malloc(sizeof(struct node));

	head->data = 45;
	head->link = second;
	second->data = 98;
	second->link = third;
	third->data = 31;
	third->link = NULL;
	menu(head);
}
```