//Single Linked List – Insertion, Deletion and Display Operations

```
#include<stdio.h>
#include<stdlib.h>

//Defining structure with node properties
struct node
{
        int no;
        struct node *next;
};

//global structure variable or node
struct node *first=0;

//Variable used to track the number of nodes in the list
int node_count=0;

//Function to display the contents of the linked list
void display()
{
        struct node *ptr1;
        printf("\n\n");
        if(first==0)
                printf("\nList is Empty");
        else
        {
                //Traverse the List
                for(ptr1=first; ptr1!=0; ptr1=ptr1->next)
                        printf("%d \t", ptr1->no);
        }
}
```

//Function to create a new list of nodes OR add nodes to the end of an existing list

```
void create()
{
        int data;
        int choice=1;

        //local structure variables or nodes. new block (nw) & temporary block used for traversal (ptr)
        struct node *ptr, *nw;

        printf("\nCreate new list of nodes or Add nodes to end of existing list");
        while (choice==1)
        {
                ++node_count;

                //DMA for new block or node
                nw = (struct node*)malloc(sizeof(struct node));
                printf("\nAddress allocated for new node: %u", nw);

                printf("\nEnter data: ");
                scanf("%d", &data);

                //Store data & address in the new block or node
                nw->no = data;
                nw->next =0;
                printf("\nBase Address:%u has Node data:%d, Node address:%u", nw, nw->no, nw->next);

                if(first==0)                                    //list is empty
                {
                        printf("\nFirst node is marked for future ref.");
                        first = nw;                             //Keep a copy of first block or node in 'first'
                }
                else
                {
                        printf("\nList traversal....");
                        for(ptr=first;ptr->next!=0;ptr=ptr->next);      //list traversal
                        ptr->next =  nw;                                //connect new node to end of list
                }
                printf("\nEnter another element(1/0): ");               //Add more nodes to the list
                scanf("%d", &choice);
        }
}
```

//Function to insert a node to the end of the Existing Linked List

```
void insertend()
{
        int data;
        struct node *ptr, *nw;

        ++node_count;

        //DMA for new block or node
        nw = (struct node*)malloc(sizeof(struct node));
        printf("\nAddress allocated for new node: %u", nw);

        printf("\nEnter data to Insert at End: ");
        scanf("%d", &data);

        //Store data & address in the new block or node
        nw->no = data;
        nw->next = 0;
        printf("\nBase Address:%u has Node data:%d, Node address:%u", nw, nw->no, nw->next);

        if(first==0)                                        //list is empty
        {
                printf("\nList empty…create a list and try again");
        }
        else
        {
                for(ptr=first;ptr->next!=0;ptr=ptr->next);  //list traversal
                ptr->next =  nw;                            //connect new node to end of list
        }
}
```

//Function to insert a node to the beginning of the Existing Linked List

```
void insertbeg()
{
        int data;
        struct node *ptr, *nw, *temp;

        ++node_count;

        //DMA for new block or node
        nw = (struct node*)malloc(sizeof(struct node));
        printf("\nAddress allocated for new node: %u", nw);

        printf("\nEnter data to Insert at Beginning: ");
        scanf("%d", &data);

        //Store data in the new block or node
        nw->no = data;

        if(first==0)                                    //list is empty
        {
                printf("\nList empty…create a list and try again");
        }
        else                                            //connect new node to beginning of list
        {
                temp=first;
                first=nw;
                first->next=temp;
        }
        printf("\nBase Address:%u has Node data:%d, Node address:%u", nw, nw->no, nw->next);
}
```

//Function to insert a node to the middle of the Existing Linked List

```
void insertmid()
{
	int data, pos, i=1;
	struct node *ptr, *nw, *temp;
	printf("\nEnter the position: ");
	scanf("%d", &pos);

	if(pos==1)
		printf("\nCannot insert at Beginning of List");
	else
	{
		ptr=first;

		//traverse the list to the given position
		while(i<pos-1)
		{
			ptr=ptr->next;
			i++;
		}

		if(ptr->next==0)                    //traversed to end-of list
			printf("\nCannot insert at End of list");
		else                                //traversed to a node in middle of list
		{
			++node_count;

			//DMA for new block or node
			nw = (struct node*)malloc(sizeof(struct node));
			printf("\nAddress allocated for new node: %u", nw);

			printf("\nEnter data to Insert in Middle: ");
			scanf("%d", &data);

			//Store data & address in the new block or node
			nw->no = data;
			nw->next = 0;
			printf("\nBase Address:%u has data:%d, address:%u", nw, nw->no, nw->next);

			//Insert new node
			temp=ptr->next;
			ptr->next=nw;
			nw->next=temp;
		}
	}
}
```

//Function to delete a node from the end of list

```
void deleteend()
{
        struct node *ptr;
        ptr=first;

        if(ptr==0)
                printf("\nList Empty");
        else if(ptr->next==0)                   //List has one node
        {
                printf("\nList has one node....");
                printf("\nNode at %p with data= %d is deleted", ptr, ptr->no);
                first=0;
                node_count=0;
        }
        else
        {
                //traverse to list
                for(ptr=first;ptr->next->next!=0;ptr=ptr->next);
                printf("\nNode at %p with data= %d is deleted", ptr->next, ptr->next->no);
                ptr->next =  0;
                --node_count;
        }
}
```


//Function to delete a node from the beginning of list

```
void deletebeg()
{
        struct node *ptr;
        ptr=first;

        if(ptr==0)
                printf("\nList Empty");
        else if(ptr->next==0)                   //List has one node
        {
                printf("\nFirst Node at %u with data= %d is deleted", ptr, ptr->no);
                first=0;
                node_count=0;
        }
        else
        {
                printf("\nFirst Node at %u with data= %d is deleted", ptr, ptr->no);
                first=ptr->next;
                --node_count;
        }
}
```

//Function to delete a node from the middle of the list

```
void deletemid()
{
	int pos, i=1;
	struct node *ptr;

	if(first==0)
		printf("\nList Empty");
	else
	{
		printf("\nEnter the position: ");
		scanf("%d", &pos);

		if(pos==1)
			printf("\nCannot delete from beginning of list");
		else
		{
			ptr=first;

			//traverse the list to the given position
			while(i<pos-1)
			{
				ptr=ptr->next;
				i++;
			}
			//traversed to end-of list
			if(ptr->next->next==0)
				printf("\nCannot delete from end of list");
			else
			{
				printf("\nNode at %u with data= %d is deleted", ptr->next, ptr->next->no);
				ptr->next=ptr->next->next;
				--node_count;
			}
		}
	}
}
```

//main function as program's entry point & from here different operations are invoked

```
int main()
{
        int option;
        L1: printf("\n\n*****************************************************");
        printf("\nSingle Linked List Operations\n");
        printf("1. Create a new List with few nodes or add nodes to end of an existing List");
        printf("\n2. Insert one node at End-of Linked List");
        printf("\n3. Insert one node at Beginning-of Linked List");
        printf("\n4. Insert one node at Middle-of Linked List");
        printf("\n5. Delete one node from End-of Linked List");
        printf("\n6. Delete one node from Beginning-of Linked List");
        printf("\n7. Delete one node from Middle-of Linked List");
        printf("\n8. Display the Single Linked List");
        printf("\nPress any other number to Stop the Program");
        printf("\n*****************************************************");

        printf("\nEnter your Choice(1-8): ");
        scanf("%d", &option);

        switch(option)
        {
                case 1:
                        create();
                        printf("\nNode Count: %d", node_count);
                        goto L1;
                case 2:
                        insertend();
                        printf("\nNode Count: %d", node_count);
                        goto L1;
                case 3:
                        insertbeg();
                        printf("\nNode Count: %d", node_count);
                        goto L1;
                case 4:
                        insertmid();
                        printf("\nNode Count: %d", node_count);
                        goto L1;
                case 5:
                        deleteend();
                        printf("\nNode Count: %d", node_count);
                        goto L1;
                case 6:
                        deletebeg();
                        printf("\nNode Count: %d", node_count);
                        goto L1;
```

```
                case 7:
                        deletemid();
                        printf("\nNode Count: %d", node_count);
                        goto L1;
                case 8:
                        display();
                        printf("\nNode Count: %d", node_count);
                        goto L1;
                default:
                        printf("\nEnding the Program...");
                        break;
        }
        return 0;
}
```