

```
/* Single Linked List Operations */
```

```
#include <iostream>
#include <stdio.h>
using namespace std;
```

```
struct node // Node structure
{
    int data;
    struct node *link;
};
struct node *root=NULL;
```

```
void append() //Function to append node in the end
{
    struct node *temp;
    temp=(struct node *)malloc(sizeof(struct node));
```

```
    cout<<"Enter Node Data"<<endl;
    cin>>temp->data;
    temp->link=NULL;
```

```
    if(root==NULL)
    {
        root=temp;
    }
    else
    {
        struct node *p;
        p=root;
        while(p->link!=NULL)
        {
            p=p->link;
        }
        p->link=temp;
    }
}
```

```
void appendatbegin() //Function to append node in the beginning
{
```

```
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
```

```
cout<<"Enter Node data"<<endl;
cin>>temp->data;
temp->link=NULL;
```

```
if(root==NULL)
{
    root=temp;
}
else
{
    temp->link=root;
    root=temp;
}
}
```

```
int length() // Function to find length of the list
{
    struct node *temp;
    int count=0;
    temp=root;
    if(temp==NULL)
        cout<<"List Empty"<<endl;
    else
    {
        while(temp!=NULL)
        {
            temp=temp->link;
            count++;
        }

        return count;
    }
}
```

```
void appendatmiddle() //Function to append a node in the middle
{
```

```
    struct node *temp,*p;
    int loc,len;
    int i=1;
```

```
    cout<<"Enter Location"<<endl; //Location after which the node will be appended
    cin>>loc;
```

```

len=length();

if(loc>len)
{
    cout<<"Invalid Location"<<endl;
    cout<<"There are "<<len<<"nodes in list"<<endl;
}
else
{
    p=root;
    while(i<loc)
    {
        p=p->link;
        i++;
    }

    temp=(struct node *)malloc(sizeof(struct node));

    cout<<"Enter Node Data"<<endl;
    cin>>temp->data;
    temp->link=NULL;

    temp->link=p->link;
    p->link=temp;

}
}

void display() //Function to display list
{
    struct node *temp;
    temp=root;
    if(root==NULL)
    {
        cout<<"List Empty"<<endl;
    }
    else
    {
        while(temp!=NULL)
        {
            cout<<temp->data<<"->";
            temp=temp->link;
        }
        cout<<"NULL";
    }
}

```

```

void deletenode() //Function to delete a node
{
    struct node *temp;
    int len,loc;
    len=length();

    cout<<"Enter Location of node to be deleted"<<endl;
    cin>>loc;

    if(loc>len)
    {
        cout<<"Invalid Location "<<endl;
        cout<<"There are "<<len<<"nodes in the list"<<endl;
    }
    else if(loc==1) //To delete First node in the list
    {
        temp=root;
        root=temp->link;
        temp->link=NULL;
        free(temp);
    }
    else //To delete both the last node and node in the middle
    {
        struct node *p, *q;
        int i=1;
        p=root;
        while(i<loc-1)
        {
            p=p->link;
            i++;
        }
        q=p->link;

        p->link=q->link;
        q->link=NULL;
        free(q);
    }
}

```

```

void deletelist() //To delete the whole list
{
    struct node *p,*q;

```

```

p=root;
while(p!=NULL)
{
    q=p->link;
    free(p);
    p=q;
}
root=NULL;
cout<<"List Deleted "<<endl;
}

```

```

int main()
{
    int ch,l=0;
    while(1)
    {
        cout<<endl<<endl<<"Linked List Operations : "<<endl;

        cout<<"1-Append"<<endl;
        cout<<"2-Append in the begining"<<endl;
        cout<<"3-Append in the middle"<<endl;
        cout<<"4-Length"<<endl;
        cout<<"5-Display List"<<endl;
        cout<<"6-Deleted node"<<endl;
        cout<<"7-Delete List"<<endl;
        cout<<"8-Exit"<<endl;

        cout<<endl<<"Enter choice"<<endl;
        cin>>ch;
        switch(ch)
        {
            case 1: append();break;
            case 2: appendatbegin();break;
            case 3: appendatmiddle();break;
            case 4: l=length();
                    cout<<"Length is "<<l<<endl;
                    break;
            case 5: display();break;
            case 6: deletenode();break;
            case 7: deletelist();break;
            case 8: exit(0); break;

            default : cout<<"Invalid Choice"<<endl;

```

}

}

}