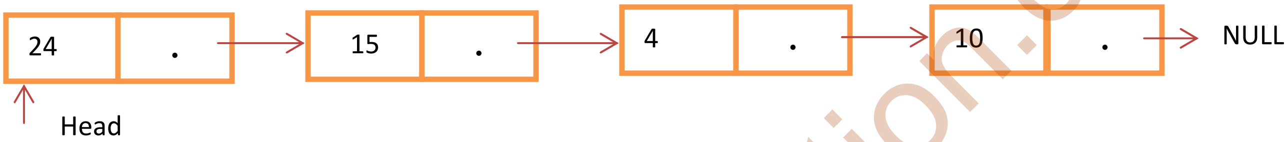
Singly Linked Lists

We have so far discussed only about singly linked list because generally linked lists means singly linked list. This list consists of number of nodes in which each node has a next pointer to the following element. The link of the last node in the list is NULL which indicates end of the list. Declaration for this type of list for integers shown below:



```
structSLnode{
        int item;
        structSLnode*  next;
};
```

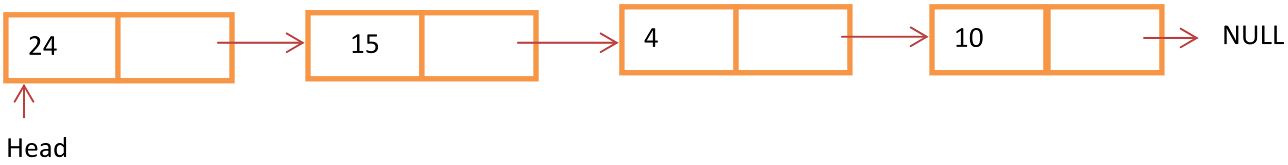
Basic Operations on a List

- Inserting an item in the list
- Deleting an item from the list
- Traversing the list

Traversing the Linked List:

Let us assume that the head points to the first node of the list. To traverse the list we do the following.

- Follow the pointer
- Display the data of the nodes or count as they are traversed.
- Stop when the next pointer points to NULL.



www.sakshieducation.com

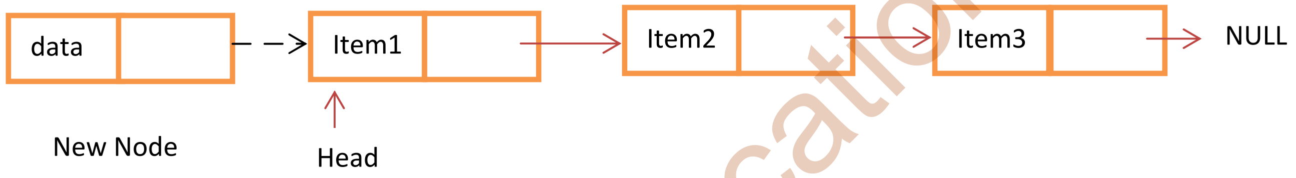
Singly Linked List Insertion

Insertion an element into singly-linked list has three cases.

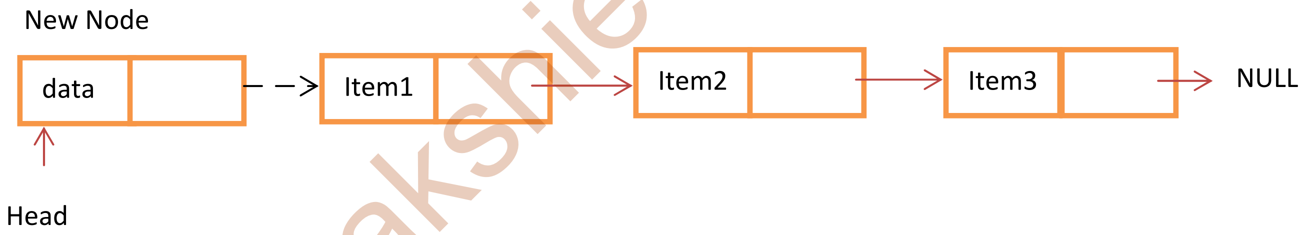
- Inserting a new node before the head (at the beginning)
- Inserting a new node after the tail (at the end of the list)
- Inserting a new node at the middle of the list (random location)

Inserting a node in singly linked list at the Beginning

- Update the next pointer of new node, to point to the current head node.

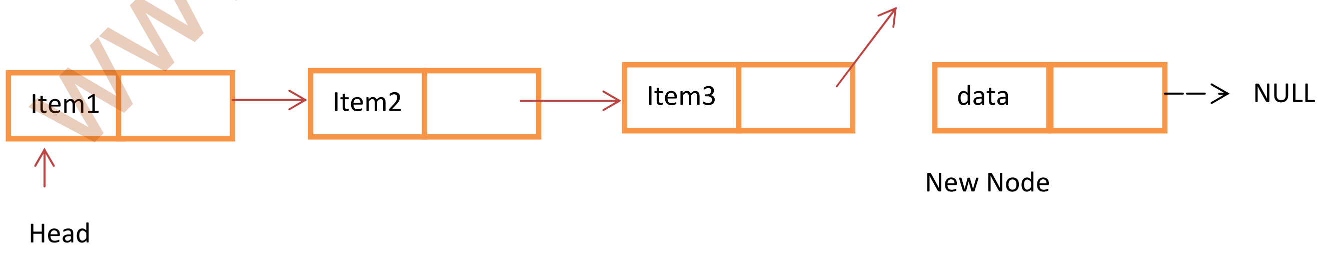


- Update the head pointer points to the new node.



Inserting a new node at the end of list

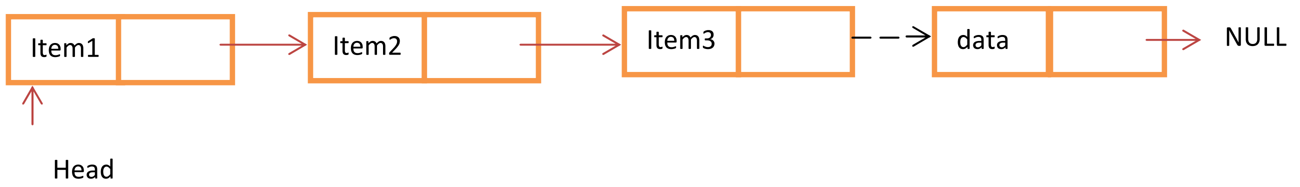
- New node next pointer points to NULL.



- Last node next pointer points new node.

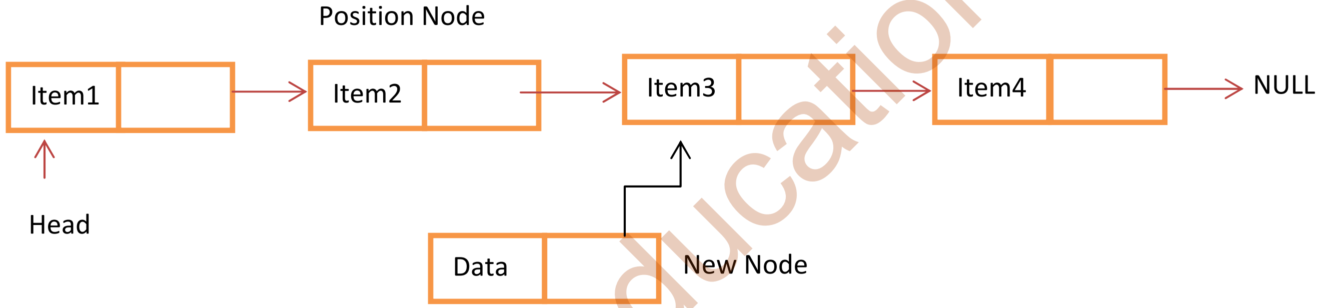
New Node

www.sakshieducation.com

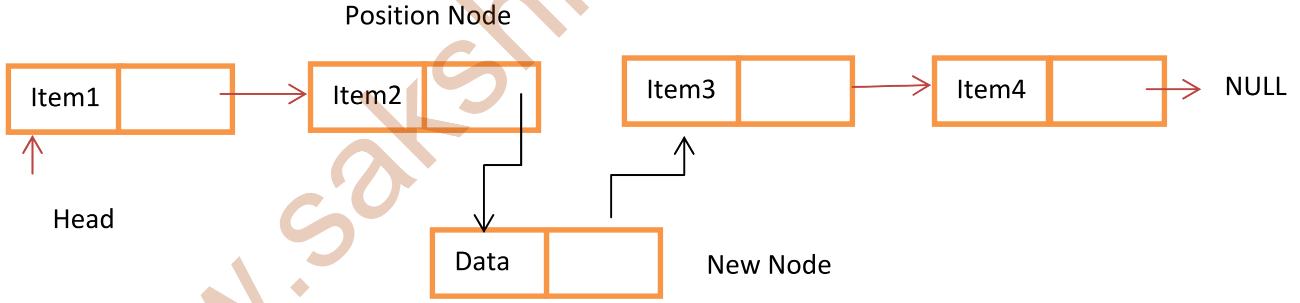


Inserting a new node at the middle of list

- If we want to add an element at 3rd position we should have to update the two pointers. First traverse the list to stop position at 2. New node points to the next node position where we want to add this node.



- Position node next pointer points to new node.



Singly Linked List Deletion

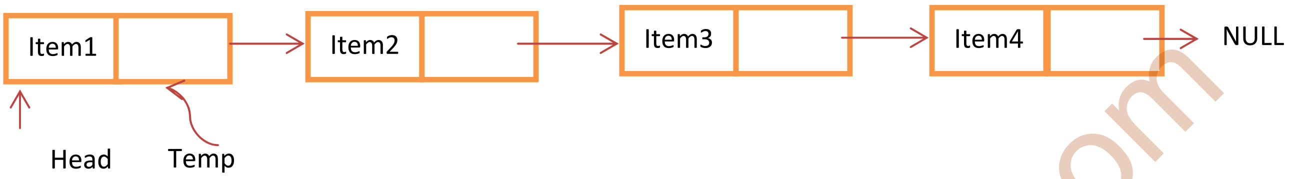
Deletion of an element from the singly-linked list has three cases.

- Deleting a new node before the head (at the beginning)
- Deleting a new node after the tail (at the end of the list)
- Deleting a new node at the middle of the list (random location)

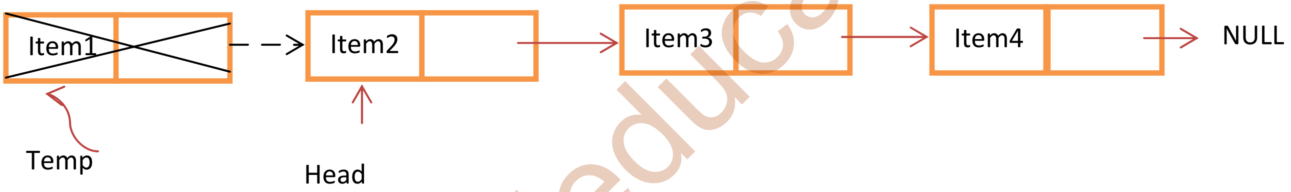
www.sakshieducation.com

Deleting a node in singly linked list at the Beginning:

- Create a temporary node which will point to the same node of that head node.

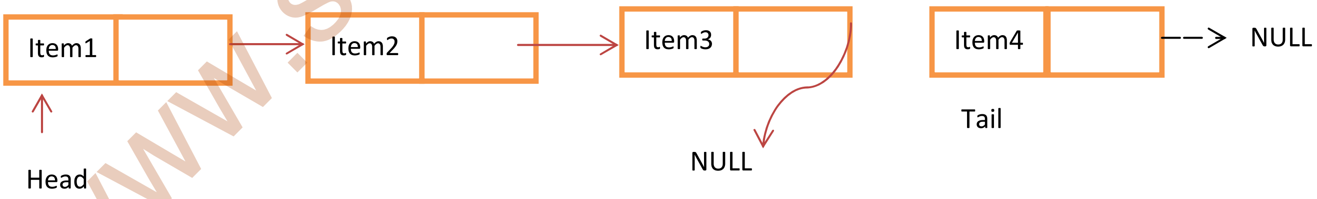


- Now update the head node pointer to the next node and free the memory of temp node.

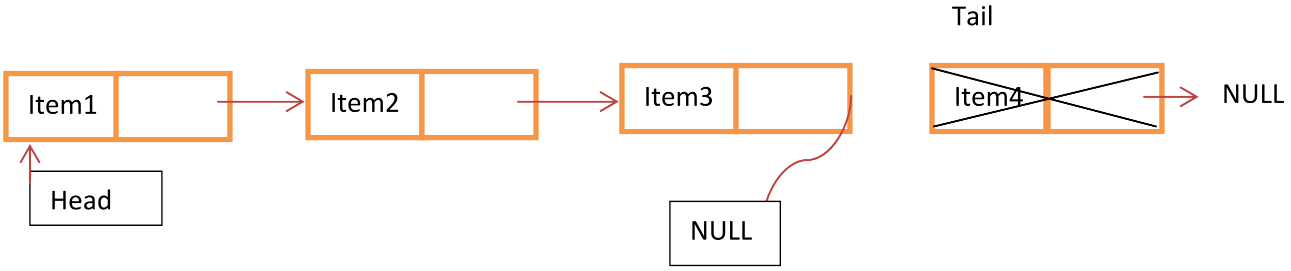


Deleting a new node at the end of list

- Traverse the list till we reach the end of list. By that time we need two pintersone pointing to tail node and other pointing to the node before tail node.



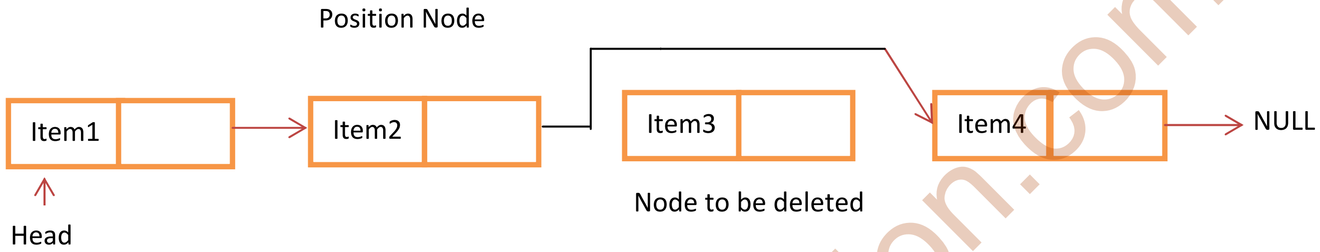
- Update the previous tail node pointer to Null and free the tail node.



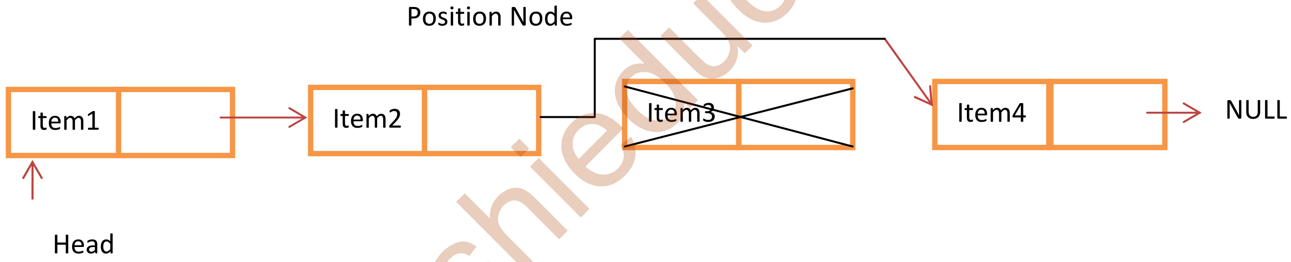
www.sakshieducation.com

Deleting a new node at the middle of list:

- If we want to delete an element at 3rd position, we should have to update the previous node next pointer to the next pointer of node deleted. First traverse the list to stop position at 2. Update the next pointer of position node.



- Position node next pointer points to next pointer of node deleted.



C Program: This is a simple program illustrate the use of singly linked list

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
//global linked list variable
struct SLL
{
int data;
struct SLL *next;//pointer to the next item
}*head; //pointer to the first entry in the list

typedefstruct SLL node;
```

www.sakshieducation.com

```
/*Function declarations and definitions*/

//Accept and store an item at beginning of the list
voidinsertAtBegning(int value)
{
node *temp;
temp=( node *)malloc(sizeof(node));//create and allocate memory to temp
node
temp->data=value;
if(head==NULL)
    {
head=temp;
head->next=NULL;

    }
else
    {
temp->next=head;
                head=temp;

    }
}
//Accept and store an item at end of the list
voidinsertAtEnd(int value)
{
node *temp,*p;

temp=(node *)malloc(sizeof( node));
temp->data=value;
if(head==NULL)
    {
head=temp;
head->next=NULL;

    }
else
    {
       p=(node *)malloc(sizeof( node));
                p=head;
```

www.sakshieducation.com

```
while(p->next!=NULL)
   {
     p=p->next;
   }

        p->next=temp;
temp->next=NULL;
  }
}

//Accept and store an item after specified item in the list
intinsertAtMiddle(int value, intloc)
{
node *temp,*p,*q;
    int i=1;
temp=(node *)malloc(sizeof(node));
temp->data=value;
if(head==NULL)
        {
temp->next=NULL;
        head=temp;
        }
else
  {
    p=head;
while(p!=NULL && p->data!=loc)
    {
      p=p->next;
    }
if(p==NULL)
    {
printf("\n%d location is not exists in list ",loc);
    }
else
    {
            q=(node *)malloc(sizeof(node));
            q=p->next;
            p->next=temp;
            temp->next=q;
         }
```

www.sakshieducation.com

```
    }

}
//delete an item at beginning of the list
voiddeleteAtBegining()
{
node *temp;
        if(head==NULL){
                printf("List is empty\n");
          }
        else{

        temp=head;
                head=head->next;
                printf("\nData deleted from list is %d \n",temp->data);
                free(temp);
                temp=NULL;
     }


}
//delete an item at end of list
voiddeleteAtEnd()
{
node *temp,*q;
        if(head==NULL){
                printf("List is empty\n");
          }
        else{
        temp=head;
                while(temp->next->next!=NULL)
                        temp=temp->next;
                q=temp->next;
                temp->next=NULL;
                printf("\nData deleted from list is %d \n",q->data);
                free(q);
          }
}
//delete specified item in the list
voiddeleteAtMiddle(intloc)
```

www.sakshieducation.com

```
{

node* p = head;
node* q = NULL;
int found = 0;
while(p != NULL)
   {
if(p->data==loc){ // you should be looking at the current node, not the
next node
found = 1;
break;
      }
                q = p;
      p = p->next;
   }
        if(found==1){
if(q==NULL)  // deleting head node
head = p->next;
else
      q->next =p->next;

        printf("\nData deleted from list is %d \n",loc);
free(p);
   }
else
printf("target not found\n");

}

void display()
{
node *t;
    t=head;
if(t==NULL)
     {
printf("List is Empty nothing to display");
     }
while(t!=NULL)
    {
printf(" %d-> ",t->data);
```

www.sakshieducation.com

```
        t=t->next;
    }
}
/*Main starts here*/
void main()
{
intvalue,i,loc,c=0;
head=NULL;//intiallize the forst pointer
printf("Select the choice of operation on link list");
printf("\n1.) insert at begning\n2.) insert at end\n3.) insert at middle");
        printf("\n4.) delete at begning\n5.) delete at end\n6.) delete at
middle");
printf("\n7.)display list\n8.)exit");
while(1)
   {
printf("\n\nenter the choice of operation you want to do ");
scanf("%d",&i);
                /*User selects particular operation to perform*/
switch(i){
case 1:
        {
printf("enter the value you want to insert in node ");
scanf("%d",&value);
insertAtBegning(value);
display();
                        break;
        }
case 2:
        {
printf("enter the value you want to insert in node at last ");
scanf("%d",&value);
insertAtEnd(value);
display();
break;
        }
case 3:
        {
printf("after which data you want to insert data ");
scanf("%d",&loc);
printf("enter the data you want to insert in list ");
```

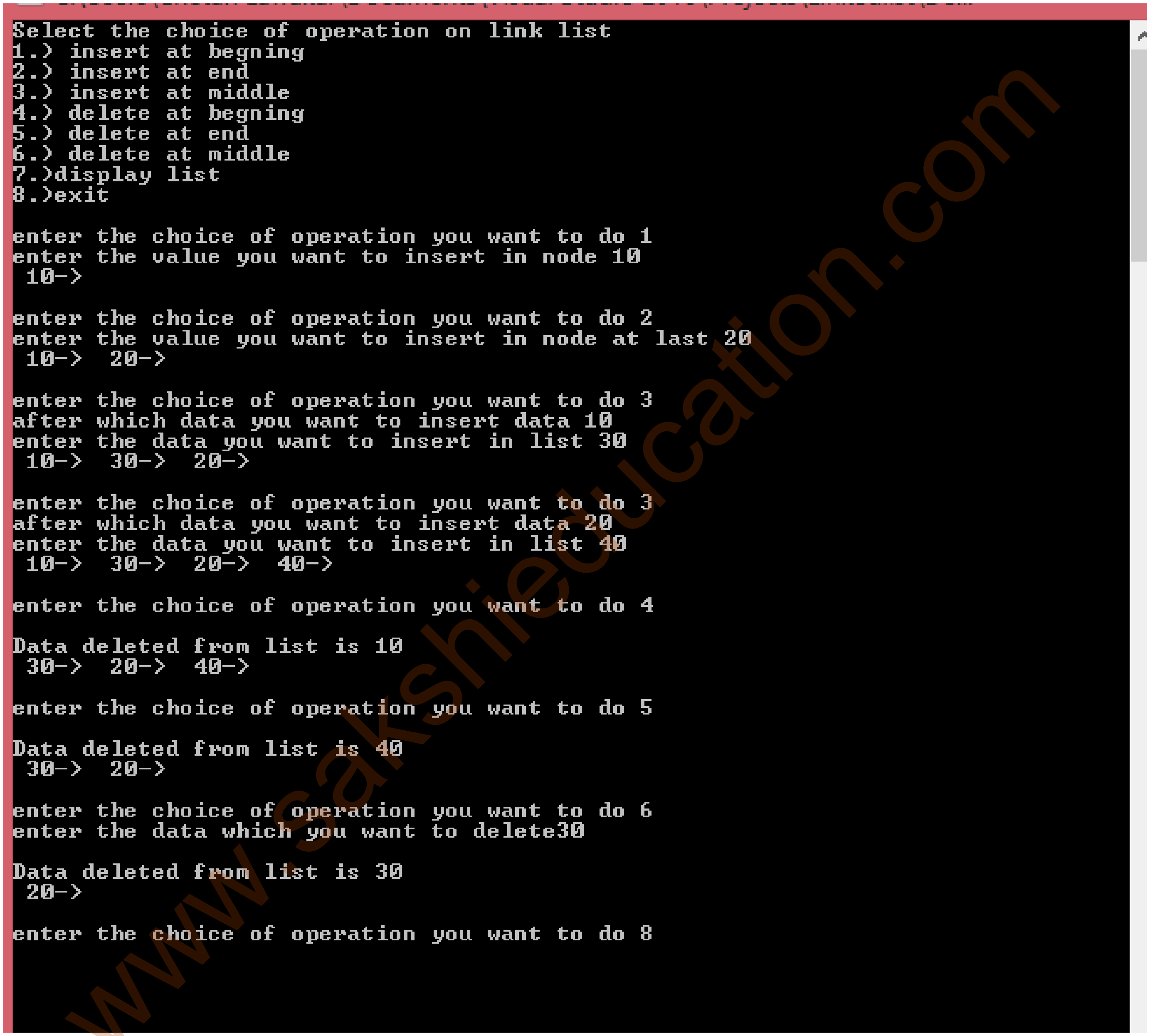
www.sakshieducation.com

```
scanf("%d",&value);
insertAtMiddle(value,loc);
display();
break;
        }
                case 4:
        {
deleteAtBegining();
display();
break;
        }
case 5:
        {
deleteAtEnd();
display();
break;
        }
case 6:
        {
        printf("enter the data which you want to delete");
        scanf("%d",&value);
                deleteAtMiddle(value);
                display();
                break;
        }
case 7:
        {
        display();
        break;
        }
case 8:
        {
exit(0);
break;
        }
        default:printf("Invalid Number\n");
                        break;
       }
   }
```

www.sakshieducation.com

```
getch();
}
```

Output:



```
Select the choice of operation on link list
1.) insert at begning
2.) insert at end
3.) insert at middle
4.) delete at begning
5.) delete at end
6.) delete at middle
7.)display list
8.)exit

enter the choice of operation you want to do 1
enter the value you want to insert in node 10
 10->

enter the choice of operation you want to do 2
enter the value you want to insert in node at last 20
 10->  20->

enter the choice of operation you want to do 3
after which data you want to insert data 10
enter the data you want to insert in list 30
 10->  30->  20->

enter the choice of operation you want to do 3
after which data you want to insert data 20
enter the data you want to insert in list 40
 10->  30->  20->  40->

enter the choice of operation you want to do 4

Data deleted from list is 10
 30->  20->  40->

enter the choice of operation you want to do 5

Data deleted from list is 40
 30->  20->

enter the choice of operation you want to do 6
enter the data which you want to delete30

Data deleted from list is 30
 20->

enter the choice of operation you want to do 8
```

www.sakshieducation.com