```
else if (temp = =tail)
{   tail = tail->prev;
    tail->next=NULL;
    temp->prev=NULL;
    free(temp);

}
else
{
    temp->prev->next=temp->next;
    temp->next->prev= temp->prev;
    free(temp);
}

}
```

~~void display();~~

Code to create doubly linked list, insert at left & right of node, delete at specific value & position & display the linked list.

```
#include <stdio.h>
#include <stdlib.h>
struct Node
{
    struct Node * prev;
    int data;
    struct Node *next;
};
struct Node *head, *tail=NULL;
void createlist(int n)
{  struct Node* newNode, *temp;
   int data;
   if (n<=0)
   { printf("number of nodes should be greater than 0\n");
     return;
   }
   for (int i=1; i<=n; i++)
   { newNode= (struct Node*) malloc (size of (struct Node));
     if (newNode==NULL)
     { printf("memory allocation fail \n");
       return;
     }
     printf("enter data :\n");
     scanf("%d", &data);
     newNode->data=data;
     newNode->next = newNode->prev= NULL;
     if (head==NULL)
         head=tail=newNode;
     else
     { tail->next= newNode;
       newNode->prev= tail;
       tail= newNode;
     }
   }
   printf(" created linked list \n");

}
void insertLeft (int data, int val)
{ struct Node * temp= head;
  Struct Node * newNode = (struct Node*) malloc (sizeof(struct Node
  newNode->data=data;
  newNode->prev= newNode->next= NULL;
  while (temp!=NULL && temp->data !=val)
      temp= temp->next;
  if (temp==NULL)
  { printf("specified value not present \n");
    free(newNode);
  }
  else {
    newNode->next= temp;
    newNode->prev = temp->prev;
    if (temp->prev!=NULL)
        temp->prev->next= newNode;
    else
        head= newNode;
    temp->prev= newNode;
  }
  printf("inserted at left of specified val \n");
}
void insertRight (int data, int val)
{ Struct Node *temp=head;
  Struct Node *newNode=(struct Node*) malloc (sizeof (struct Node));
  newNode->data = data;
  newNode->prev = newNode->next = NULL;
```

```
while(temp!
    temp=temp->next;
if (temp==NULL)
{ printf("specified value not present\n");
  free(newNode);
}
else{
  newNode->next=temp->next;
  newNode->prev=temp;
  if (temp->next!=NULL)
     temp->next->prev=newNode;
  else
     tail=newNode;
  temp->next=newNode;
}
printf(" inserted at right of specified value\n");
}
void deletespecific(int val)
{ struct Node *temp=head;
  if (head==NULL)
  { printf(" linked list not present\n");
    return;
  }
  while(temp!=NULL && temp->data!=val)
      temp=temp->data;
  if(temp==NULL)
  { printf("specified value not present\n");
    return;
  }
  if(temp==head)
  { ~~temp~~ head=head->next;
    if(head!=NULL)
       head->prev=NULL;
    else
       tail=NULL;
    free(temp);
  }
  else if(temp==tail)
  { tail=tail->prev;
    tail->next=NULL;
    temp->prev=NULL;
    free(temp);
  }
```

```
  else{
     temp->prev->next=temp->next;
     temp->next->prev=temp->prev;
     free(temp);
  }
}
void deleteAtPosition(int pos)
{ for(int i=1; i<pos && temp!=NULL; i++)
     temp=temp->next;
  if (temp==NULL)
     printf(" position out of range\n");
  else
  { temp->prev->next=temp->next;
    if (temp->next!=NULL)
       temp->next->prev=temp->prev;
    else
       tail=temp->prev;
    free(temp);
    printf("deleted at position %d\n", pos);
  }
}
void display()
void main()
{ int ch, n, val, pos, data;
  printf(" 1.create doubly linked list, 2.insert at left,
   3.insert at right, 4.delete at specific val, 5.delete at
   specific pos, 6.display\n");
  do{ printf(" enter choice: \n");
      scanf("%d", &ch);
      switch(ch){
         case 1: printf("enter number of nodes: ");
                 scanf("%d", &n);
                 createList(n);
                 break;
         case 2: printf(" enter data\n");
                 scanf("%d", &data);
                 printf("enter value\n");
                 scanf("%d", &val);
```

```
        insertLeft
        break;
case 3: printf("enter data:\n");
        scanf("%d",&data);
        printf("enter value:\n");
        scanf("%d", &val);
        insertRight(data, val);
        break;
case 4: printf("enter specific value:\n");
        scanf("%d",&val);
        deleteSpecific(val);
        break;
case 5: printf("enter specific pos:\n");
        scanf("%d", &pos);
        deleteAtposition(pos);
        break;
case 6: { display();
        break;}
default:
        printf("invalid choice\n");
}
}while(ch!=0);
return0;
}
```

Output:

```
1.create linked list ; 2. insert at left, 3.insert right, 4.delete value
5.delete atpos , 6.display.
enter choice:1
enter number of nodes:3
enter data: 10
enter data:20
enter data:30
created linked list
enter choice:2
enter data: 40
enter value: 20
inserted at left
enter choice:3
enter data: 50
enter value:30
inserted at right
enter choice: 4
enter specific value:20
enter choice:5
enter specific pos:3
deleted at position3
enter choice:
6
10 <-> 40 -> 50 <->
enter choice: 0
invalid choice.
```

Leet code

```
bool hasCycle(struct ListNode *head)
{   if (head == NULL || head->next==NULL)
        return false;
    struct ListNode *slow = head;
    struct ListNode *fast = head;
    while (fast != NULL && fast->next != NULL)
    {   slow = slow->next;
        fast = fast->next->next;
        if (slow == fast)
            return true;
    }
    return false;
}
```

Output:

```
Input:   head = [3,2,0,-4]
         pos = 1
Output : true
```

Both o/p seen
8