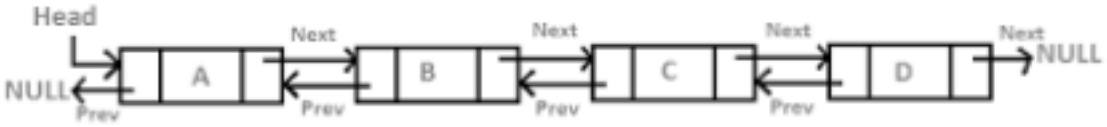
GeeksMan
Linked List
Lesson 2

Doubly Linked List

A Doubly Linked List (DLL) contains an extra pointer, typically called previous pointer, together with next pointer and data which are there in singly linked list.

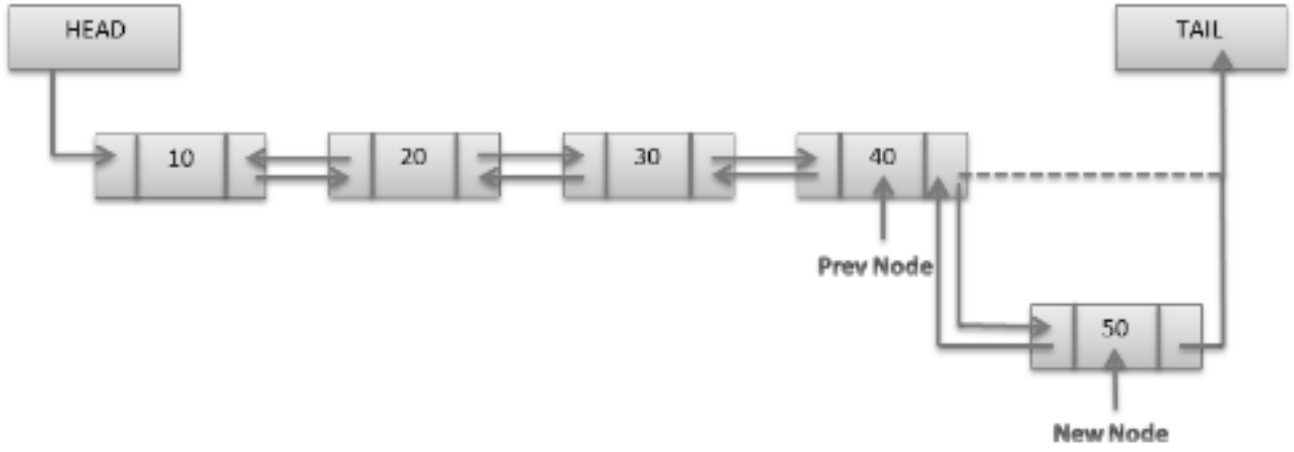


Representation

```
1.  struct Node
2.  {
3.          int data;
4.          struct Node* next;
5.          struct Node* prev;
6.  };
```

Doubly linked list Insertion at given position

Given a doubly-linked list, a position p, and an integer x. The task is to add a new node with value x at the position just after pth node in the doubly linked list.



Input:

LinkedList: 2<->4<->5

p = 2, x = 6

Output: 2 4 5 6

Explanation: p = 2, and x = 6. So, 6 is inserted after p, i.e, at position 3

Solution :

```
#include <bits/stdc++.h>
using namespace std;
struct Node
{
 int data;
 struct Node *next;
 struct Node *prev;
 Node(int x)
 {
    data = x;
    next = prev = NULL;
 }
};
void addNode(Node *head, int pos, int data);
Node *insert(Node *head, int x)
{
  if (head == NULL)
  {
     return new Node(x);
  }
  Node *n = new Node(x);

  head->next = n;
  n->prev = head;
  head = n;
  return head;
}

```

```
29.void printList(Node *head)
30.{
31.   Node *temp=head;
32. if (temp != NULL)
33.{
34.
35. while (temp->next!=NULL)
36.   temp=temp->next;
37. while (temp->prev!=NULL)
38.  temp = temp->prev;
39. }
40. while (temp != NULL)
41. {
42.     printf("%d ",temp->data);
43.     temp=temp->next;
44. }
45. cout << endl;
46.}
47.int main()
48.{
49. int t;
50. scanf("%d",&t);
51.  while(t--) {
52. Node *head = NULL;
53. Node *root = NULL;
54. int n;
55. scanf("%d",&n);
56. for(int i=0;i<n;i++){
57.    int x;
58.    scanf("%d",&x);
```

```
59.    head = insert(head, x);
60.    if(root==NULL) root = head;
61. }
62. head = root;
63. int pos,data;
64. cin>>pos>>data;
65. addNode(head, pos, data);
66. printList(head);
67. }
68. return 0;
69.}
70.void addNode(Node *head, int pos, int data)
71.{
72.   int i = 0;
73.   while(i<pos)
74.   {
75.       head = head->next;
76.       i++;
77.   }
78.   Node *newnode = new Node(data);
79.   newnode->next = head->next;
80.   head->next = newnode;
81.   newnode->prev = head;
82.}
83.
```

Delete node in Doubly Linked List

Given a doubly linked list and a position. The task is to delete a node from a given position in a doubly linked list.

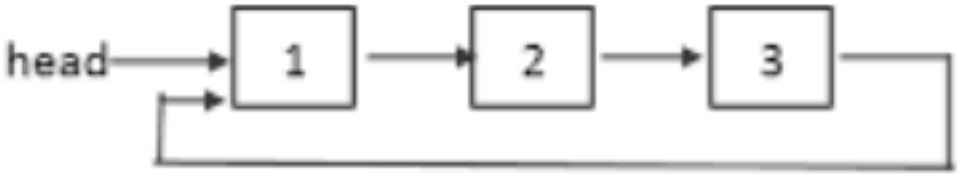
```
1.  Node* deleteNode(Node *head_ref, int x)
2.  {
3.     int i = 1;
4.     Node *temp ;
5.     temp = (head_ref);
6.     while(i<=x)
7.     {
8.        if(i == x)
9.        {
10.          if(temp->next==NULL)
11.           {
12.              (temp->prev)->next=NULL;
13.           }
14.           else if(temp->prev==NULL)
15.           {
16.              temp=temp->next;
17.              temp->prev=NULL;
18.              head_ref=temp;
19.           }
20.          else
21.          {
22.             temp->prev->next = temp->next;
23.             (temp->next)->prev = temp-> prev;
24.          }
```

```
25.      }
26.      else
27.      {
28.          temp = temp->next;
29.      }
30.      i++;
31.   }
32.   return head_ref;
33. //Your code here
34.}
```

Circular Linked List

Circular linked list is a linked list where all nodes are connected to form a circle. There is no NULL at the end. A circular linked list can be a singly circular linked list or doubly circular linked list.

Singly Linked List



Circular Linked List

Check If Circular Linked List

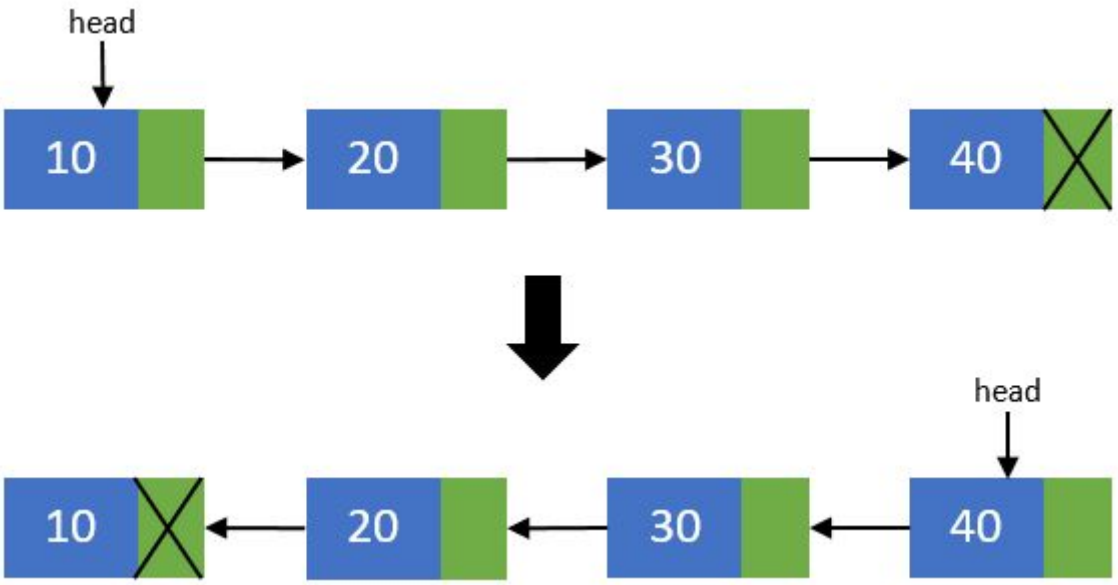
Given a singly linked list, find if the linked list is circular or not. A linked list is called circular if it is not NULL terminated and all nodes are connected in the form of a cycle. An empty linked list is considered as circular.

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.  #include<iostream>
4.  using namespace std;
5.  struct Node
6.  {
7.      int data;
8.      struct Node* next;
9.
10.    Node(int x){
11.        data = x;
12.        next = NULL;
13.    }
14.
15. };
16.   bool isCircular(struct Node *head);
17.  int main()
18. {
19.  int T,i,n,l,k;
20.
21.    cin>>T;
22.
23.    while(T--){
24.        cin>>n>>k;
```

```
25.             Node *head=NULL, *tail = NULL;
26.             int x;
27.             cin >> x;
28.             head = new Node(x);
29.             tail = head;
30.             for(int i=0;i<n-1;i++)
31.             {
32.                 cin>>x;
33.                 tail -> next = new Node(x);
34.                 tail = tail -> next;
35.             }
36.       if (k==1 && n >= 1)
37.             tail->next = head;
38.        printf("%d\n", isCircular(head));
39.   }
40.   return 0;
41. }
42.bool isCircular(Node *head)
43.{
44.   struct Node *temp;
45.   temp = head;
46.     while(temp->next != NULL && temp->next != head)
47.   {
48.       temp = temp->next;
49.   }
50.   if(temp->next == head)
51.       return true;
52.   return false;
53.  // Your code here
54.}
```

Reverse a linked list

Given a linked list of N nodes. The task is to reverse this list.



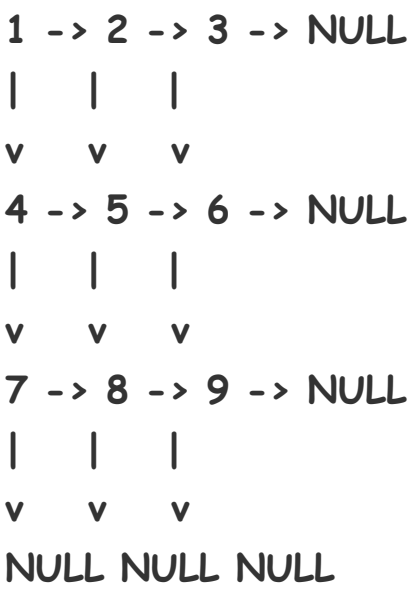

```
1.  struct Node* reverseList(struct Node *head)
2.  {
3.     struct Node *curr = head;
4.     struct Node *prevn = NULL , *nextn = head;
5.     while(nextn != NULL )
6.     {
7.        nextn = curr->next;
8.        curr->next = prevn;
9.        prevn = curr;
10.       curr = nextn;
11.    }
12.    return prevn;
13. }
```

Linked List Matrix

Given a Matrix **mat** of **N*N** size, the task is to complete the function **constructLinkedMatrix()**, that constructs a 2D linked list representation of the given matrix.

Input : 2D matrix
1 2 3
4 5 6
7 8 9

Output :



occurence-of-an-integer-in-a-linked-list
count-pairs-whose-sum-is-equal-to-x
circular-linked-list-traversal
rotate-doubly-linked-list-by-p-nodes
reverse-a-doubly-linked-list