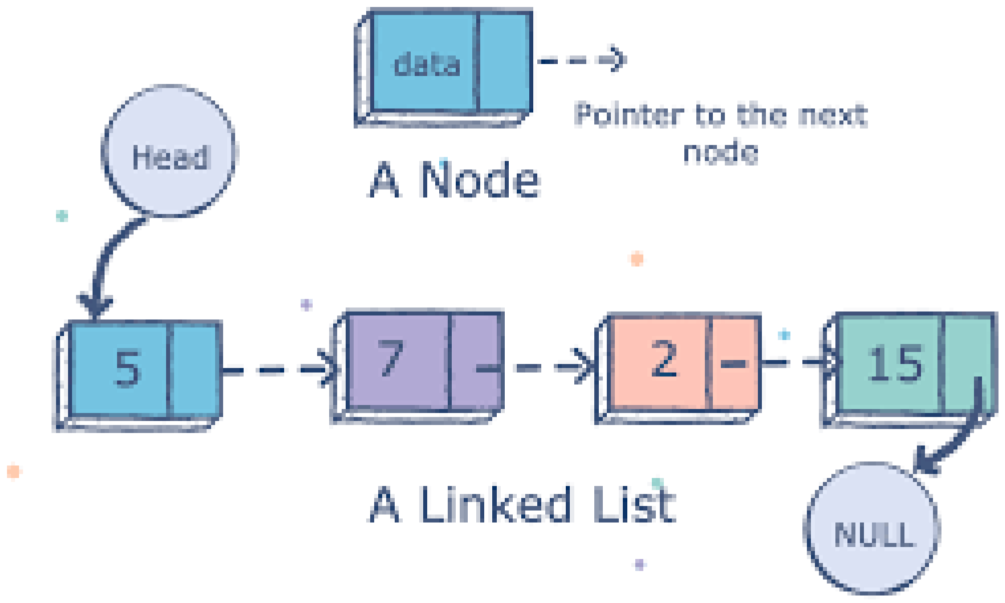
GeeksMan
Linked List
Lesson 1

GeeksMan

We code therefore we are

Linked List

In computer science, a **linked list** is a linear collection of data elements whose order is not given by their physical placement in memory. Instead, each element points to the next. It is a data structure consisting of a collection of nodes which together represent a sequence. In its most basic form, each node contains: data, and a reference (in other words, a *link*) to the next node in the sequence.



Why Not arrays??

(1) The size of the arrays is fixed : we have to specify the number of elements in the array at the compile time .

(2) Inserting a new element and deleting an element in an array of elements is expensive.

So Linked list provides the following two **advantages** over arrays :

1) Dynamic size

2) Ease of insertion/deletion

Drawbacks of Linked List:

1) Random access is not allowed. We have to access elements sequentially starting from the first node.
2) Extra memory space for a pointer is required with each element of the list.

Representation:

A linked list is represented by a pointer to the first node of the linked list. The first node is called the head. If the linked list is empty, then the value of the head is NULL.

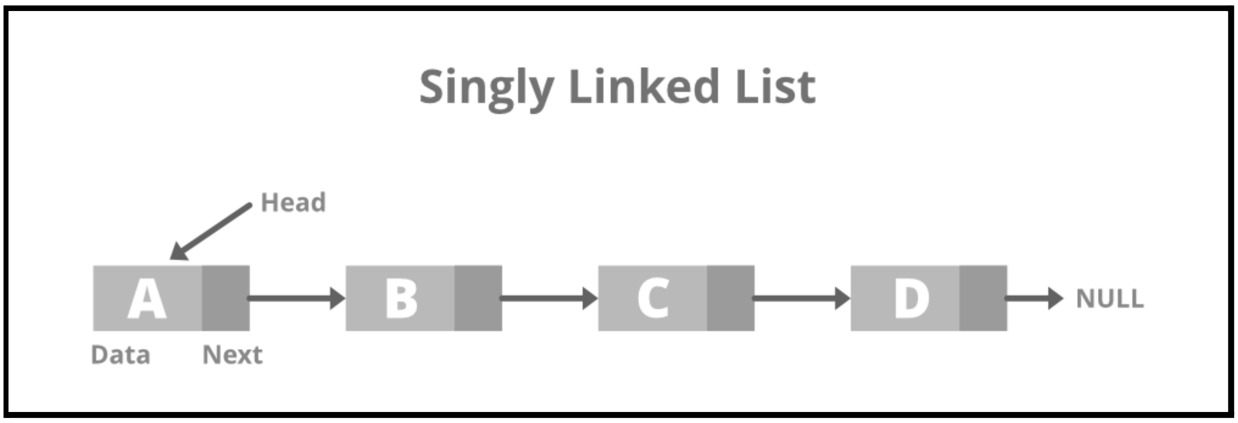
Each node in a list consists of at least two parts:
1) data
2) Pointer (Or Reference) to the next node

```
struct Node {
        int data;
        struct Node* next;
};
```

Construct a Linked List : 1->2->3

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

struct Node {
	int data;
	struct Node* next;
};
int main()
{
	struct Node* head = NULL;
	struct Node* second = NULL;
	struct Node* third = NULL;

	head = (struct Node*)malloc(sizeof(struct Node));
	second = (struct Node*)malloc(sizeof(struct Node));
	third = (struct Node*)malloc(sizeof(struct Node));

	head->data = 1;
	head->next = second;

	second->data = 2;
	second->next = third;

	third->data = 3;
	third->next = NULL;

	return 0;
}
```

Traversal of a linked list



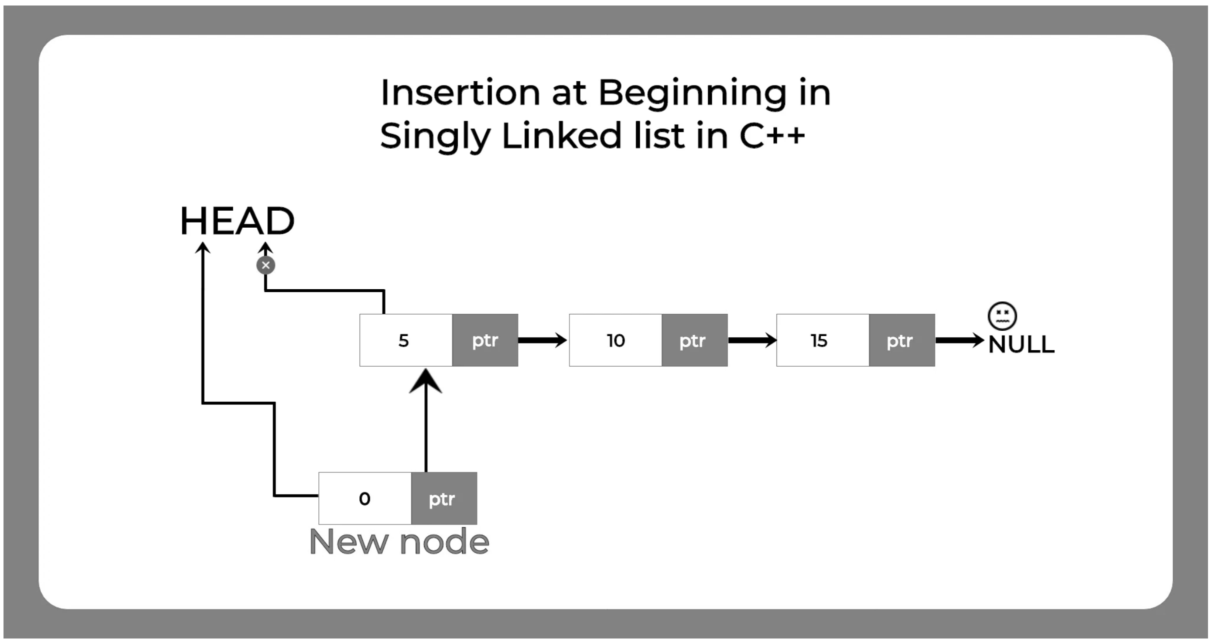
```c
#include <stdio.h>
#include <stdlib.h>

struct Node
{
        int data;
        struct Node* next;
};

void printList(struct Node* n)
{
        while (n != NULL) {
                printf(" %d ", n->data);
                n = n->next;
        }
}
```

```

int main()
{
        struct Node* head = NULL;
        struct Node* second = NULL;
        struct Node* third = NULL;

        head = (struct Node*)malloc(sizeof(struct Node));
        second = (struct Node*)malloc(sizeof(struct Node));
        third = (struct Node*)malloc(sizeof(struct Node));

        head->data = 1;
        head->next = second;

        second->data = 2;
        second->next = third;

        third->data = 3;
        third->next = NULL;

        printList(head);

        return 0;
}
```

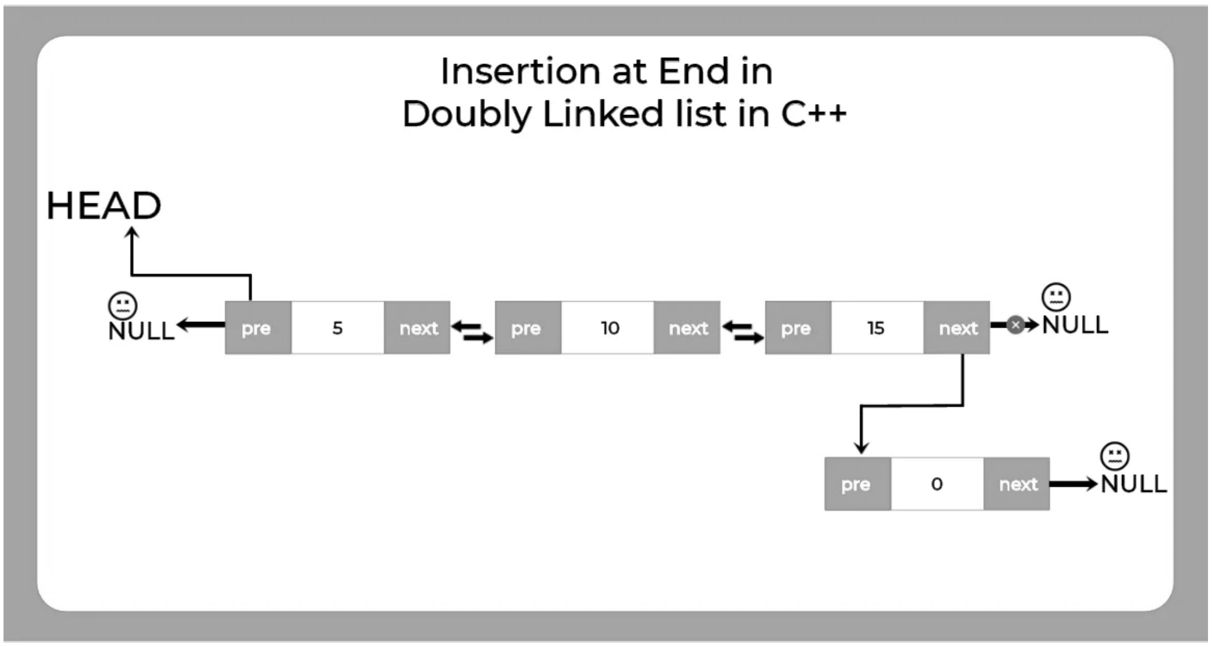
Count Nodes in a Linked List

```cpp
#include <bits/stdc++.h>
using namespace std;

struct Node
{
    int data;
    struct Node* next;

    Node(int x){
        data = x;
        next = NULL;
    }
};

int getCount(struct Node* head);

int main()
{
    int t;
    cin>>t;
    while(t--)
    {
        int n;
        cin>>n;

        int data;
        cin>>data;
        struct Node *head = new Node(data);
        struct Node *tail = head;
        for (int i = 0; i < n-1; ++i)
        {
            cin>>data;
            tail->next = new Node(data);
            tail = tail->next;
        }
        cout << getCount(head) << endl;
    }
    return 0;
}
int getCount(struct Node* head)
{
    int count = 1;
    struct Node* temp;
    temp = head;
    while(temp->next != NULL )
    {
        count++;
        temp = temp->next;
    }
    return count;
    //Code here

}
```

Insert a Node In a Linked List

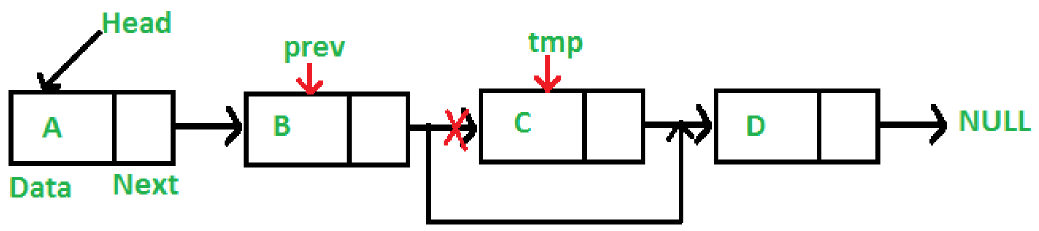


```
Node *insertAtBegining(Node *head, int newData)
{
   struct Node *newnode;
   newnode = new Node(newData);
   newnode->next = head;
   return newnode;
 }
```



```
Node *insertAtEnd(Node *head, int newData)
{
   struct Node *newnode;
   newnode = new Node(newData);
    if(head==NULL)
  {
      return newnode;
  }
   struct Node *temp;
   temp = head;
   while(temp->next != NULL)
   {
      temp = temp->next;
   }
   temp->next = newnode;
    return head;

```

Delete Node In a Linked List



```
Node* deleteNode(Node *head,int x)
{
   Node *temp = head , *ptr;
   if(temp->data == x)
   {
      head = temp->next;
      return head;
   }
   while(temp->next)
   {
      ptr = temp;
      if(temp->next->data == x)
      {
         temp->next = temp->next->next;
         return head;
      }
      temp = temp->next;
   }
   if(temp->data == x)
      ptr->next = NULL;
      return head;
   }
```

Count-nodes-of-linked-list
linked-list-length-even-or-odd=
identical-linked-lists
print-linked-list-elements
modular-node
Finding-middle-element-in-a-linked-list
Insert-in-middle-of-linked-list
node-at-a-given-index-in-linked-list
find-nk-th-node-in-linked-list
linked-list-insertion-