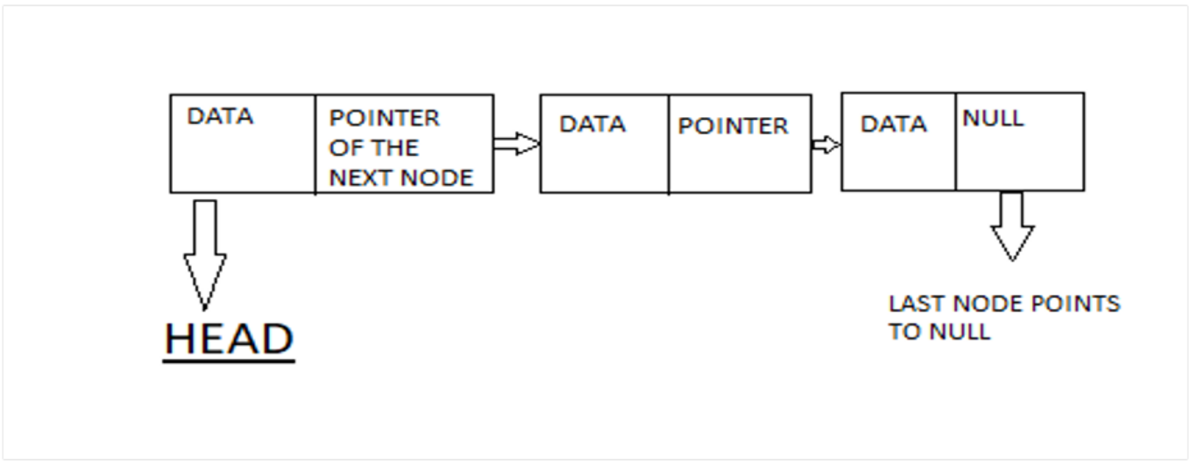
LINKED LIST

What is linked list?

→ Linked list is a linear data structure that consists of series of connected nodes.

- Singly Linked list
- Doubly Linked list
- Circular linked list



Address of first node is called HEAD and the last node pointer points to none.

How is each node represented?

```
 struct nodes
{
  Int data;           //stores the data of the node
  struct node *next; //pointer pointing to the next node
};
```

Why linked list?

→ Applications:

- Deployed on stacks and queue
- Hash tables and graphs

- Dynamic memory allocation

Implementation:

```c
//Program to define a linked list having 3 nodes
#include <stdio.h>
//Declaring a node
struct node
{
  int data;          //value of node
  struct node *next; //pointer pointing to next node
};
// print the linked list value
void printLinkedlist(struct node *l)
{
  while (l != NULL)
  {
    printf("%d ", l->data);
    l = l->next;
  }
}
int main()
{
  //Initialize nodes
  struct node *head;
  struct node *first = NULL;
  struct node *second = NULL;
  struct node *third = NULL;
  //Dynamic memory allocation
  first = malloc(sizeof(struct node));
  second = malloc(sizeof(struct node));
  third = malloc(sizeof(struct node));
  //Assign data value
  first->data = 3;
  second->data = 1;
  third->data = 4;
  //pointers pointing to next node
  first->next = second;
  second->next = third;
  third->next = NULL; //last node points to none
  //printing linked list
  printLinkedlist(first); //head is the first node
}
```

Adding node and traversing through linked list:

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

struct Node{
    int data;
    struct Node *next;
};
void addAtBegin(struct Node **start, int d){
    struct Node *t;
    t = (struct Node *)malloc(sizeof(struct Node));
    t->data = d;
    t->next =  *start;
    *start = t;
}
void traverse(struct Node *start){
    if(start == NULL){
        printf("\nEmpty list.");
    }
    while(start != NULL){
        printf("%d ", start->data);
        start = start->next;
    }

}
int main()
{
    struct Node *linkedList;
    linkedList = NULL;
    addAtBegin(&linkedList, 10);
    addAtBegin(&linkedList, 20);
    addAtBegin(&linkedList, 30);
    addAtBegin(&linkedList, 40);
    addAtBegin(&linkedList, 50);
    traverse(linkedList);
    return 0;
}
```

What is Doubly linked list?

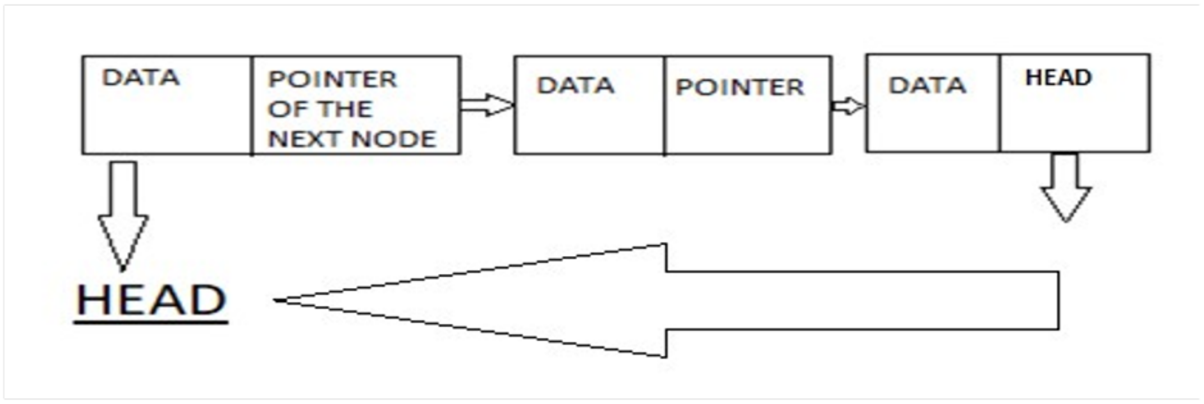
→ In doubly linked list we have a pointer that points to the previous node which allows the user to navigation both back and forth.

Implementation:

```c
//Program to define a linked list having 3 nodes and assignng
them values and printing them.
#include <stdio.h>
//Declaring a node
struct node
{
  int data;               //value of node
  struct node *next;      //pointer pointing to next node
  struct node *previous; //pointer pointing to previous node
};
int main()
{
  //Initialize nodes
  struct node *head;
  struct node *first = NULL;
  struct node *second = NULL;
  struct node *third = NULL;
  //Dynamic memory allocation
  first = malloc(sizeof(struct node));
  second = malloc(sizeof(struct node));
  third = malloc(sizeof(struct node));
  //Assign data value
  first->data = 3;
  second->data = 1;
  third->data = 4;
  //pointers pointing to next node
  first->next = second;
  first->previous = NULL;
  second->next = third;
  second->previous = first;
  third->next = NULL; //last node points to none
  third->previous = second;
}
```

Circular linked list

→ In circular linked list the pointer of the last node points to the first node i.e. HEAD.


DATA
POINTER OF THE NEXT NODE
DATA
POINTER
DATA
HEAD
HEAD