(Data structure using C - 22317)

Unit IV : Linked List

(weightage: 16 marks)

Pranjal Save (SY comps)

Introduction

* Linked List is a collection of data elements stored in such a manner that each element points at the next element in the list.
* Elements of a linked list are also referred as nodes.

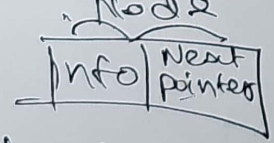
Questions

Q) Basic Terminologies of Linked list (Explain)
i) Node ii) Next pointer iii) NULL Pointer
iv) Empty List (6 marks)

5.1 : Introduction to Linked List Terminologies - Node, Address, pointer, Information field / Data field, Next pointer, Null pointer, Empty List.

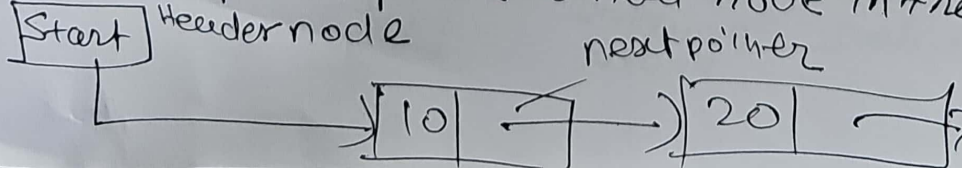
Node

Each data element in linked list is represented as a node.
Node contains two parts - one is info (data) and other is next pointer (address).
Info part stores data and next pointer stores address of next node.



Next pointer

It is the pointer that holds address of next node in the list.
i.e next pointer points to next node in the list.

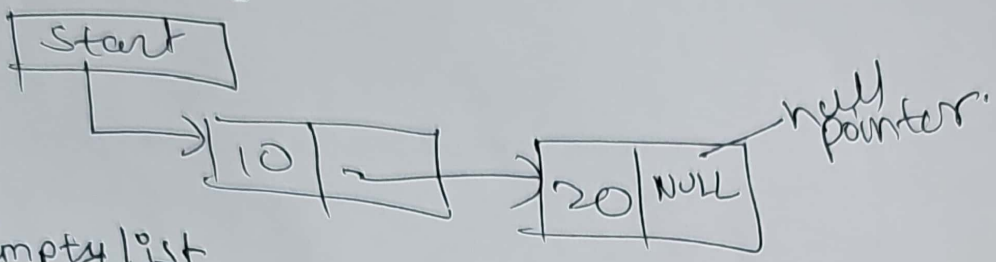


NULL pointer

It is that pointer that does not hold any memory address i.e it is pointing nothing.

The last element of list contains NULL pointer to specify end of list.

Header node



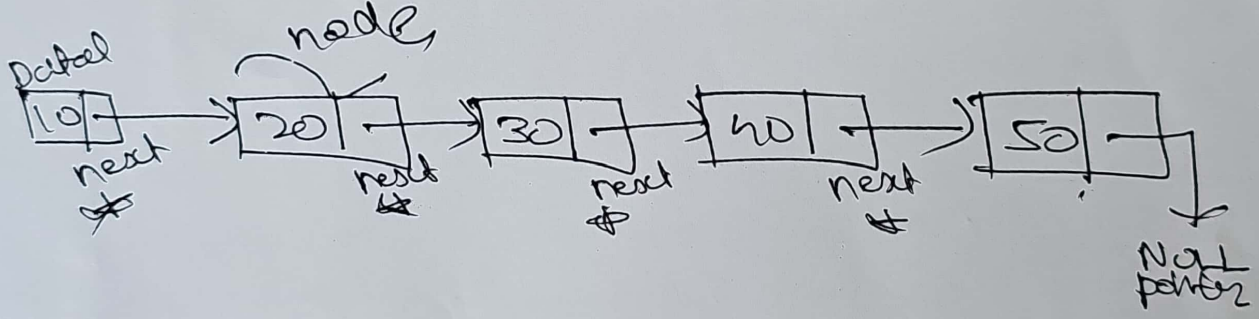
Empty list

Each linked list has a header node.

When header node contains NULL value, then that list is said to be empty ~~list~~ list.

Header node

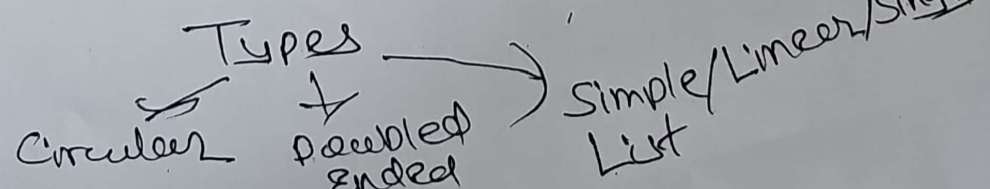
Linked List representation. 10, 20, 30, 40, 50

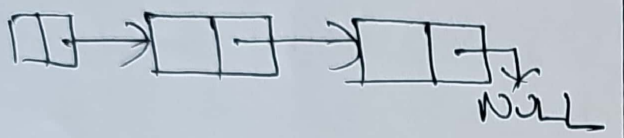
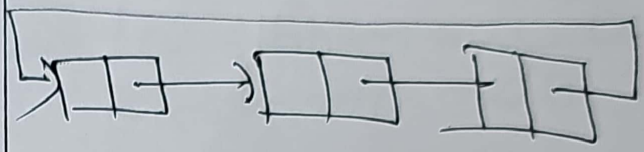


5.2 Types of lists: Linear List, Circular list

Q// Write advantage of single linked List over array. (4m).

Q// Compare linear list & Circular list.



Linear List	Circular list
Linear/simple list is a list where last pointer is null not connected to first node.	Circular list the last node having pointer stores address of head node.
Representation 	Representation 
It contains NULL pointer	It does not contain null pointer.
Singly linked list.	
Next pointer and last pointer NULL.	Next pointer at head last pointer.
It is simple.	It is complex
Slower.	faster

3.2 Operations on singly linked list.

Traversing a singly linked list, Searching key in linked list, Inserting a new node in linked list, Deletion a node from a linked list.

Questions

Q1 Write algorithm to traverse linked list.

Q2 Algorithm to count no of nodes in SLL.

Q3 Write an algorithm insert/delete/element ~~at also middle at be~~ at beginning of list. also intermediate

Insert at begining

1. IF PTR = NULL then, Goto step 7 or else Step 2.
2. Set New_Node = pointer
3. Set pointer = pointer → next.
4. Set new_node → data = value.
5. Set New_node → next = head.
6. Step Set head = New_Node.
7. Stop.

Deletetion at Begining

1. Start
2. If head = NULL (End) or else step 3.
3. Create a new node temp of node type and initialize with head.
4. Set head = head → next
5. Set temp → next = NULL
6. Delete temp.
7. Return head.

...ting at Between / middle of linked list

1. ...llocate a new node.
2. ...insert new element.
3. Goto node that should follow the one to add.
4. Having that node point to new node.
5. Have a new node point to node next to the node found.

Deletion at Begining of Singly linked list

Step 1: If Head = NULL. (Exit)
else Step 2

Step 2: Set PTR = HEAD.

Step 3: Set HEAD = HEAD → NEXT.

Step 4: FREE PTR.

Step 5: Stop.

Deletion at End of Singly linked list

1. If Head = NULL stop.
2. SET PTR = HEAD.
3. Repeat step 4 & 5 while PTR → NEXT != NULL
4. SET PREPTR = PTR.
5. Set PTR = PTR → NEXT.
6. Set PREPTR → NEXT = NULL.
7. Free PTR
8. Stop.

Traverse a singly linked list.

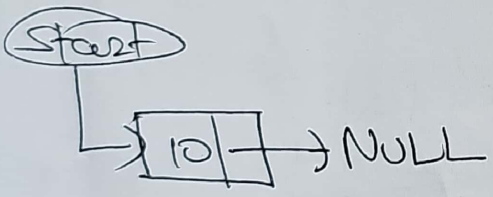
1. Set PTR = HEAD.
2. If PTR = NULL.
   "LIST IS EMPTY"
   exit.
   Else Step 3.
3. Repeat step 4 & 5 until (PTR) = NULL
4. PRINT PTR → DATA.
5. PTR = PTR → NEXT (End of loop).
6. Exit.

Create singly linked list 10, 20, 30, 40, 50.

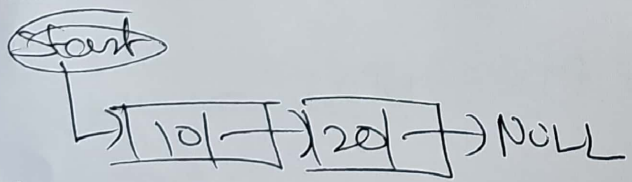
Step by step.

Step 1: Empty list

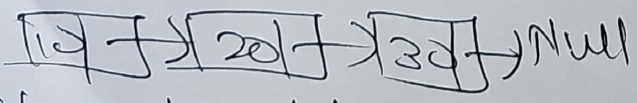
Step 2: Insert 10



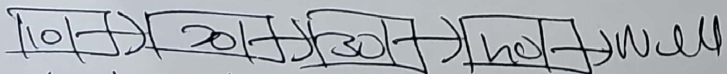
Step 3: 20 Insert



Step 4: Insert 30



Step 5: Insert 40



Step 6: Insert 50

Search 30

Step 1: check if PTR=NULL. "no"

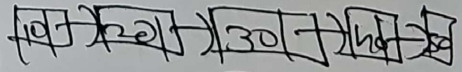
Step 2

30
no found.

Step 3

30
no found.

Step 4



30
found.

Step 5: Stop.