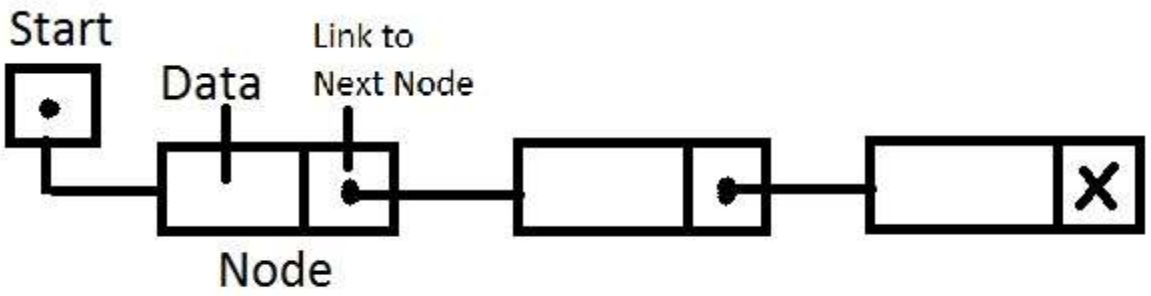
MODULE 3
LINKED LIST

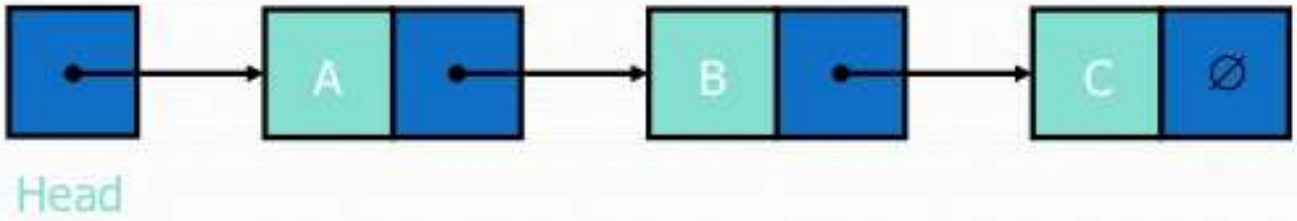
LINKED LIST Definition:

- A **linked list** is a linear data structure where each element is a separate object.
- **Linked list** elements are not stored at contiguous location; the elements are **linked** using pointers.
- LINKED LIST is a collection of data elements called nodes where the linear order is given by means of pointer.
- Each node of a **list** is made up of two items - the data and a reference to the next node.
- The last node has a reference to null.

Representation of a node:

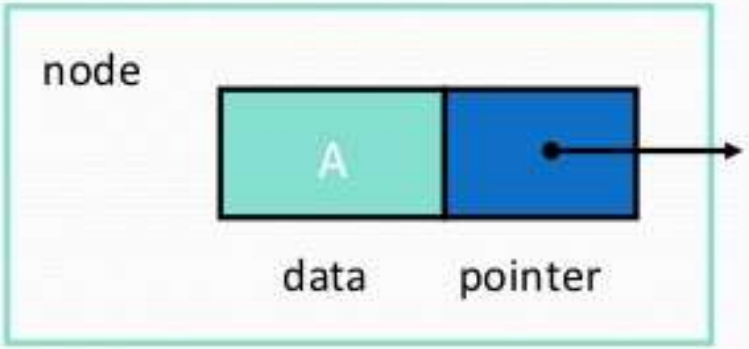


Linked Lists

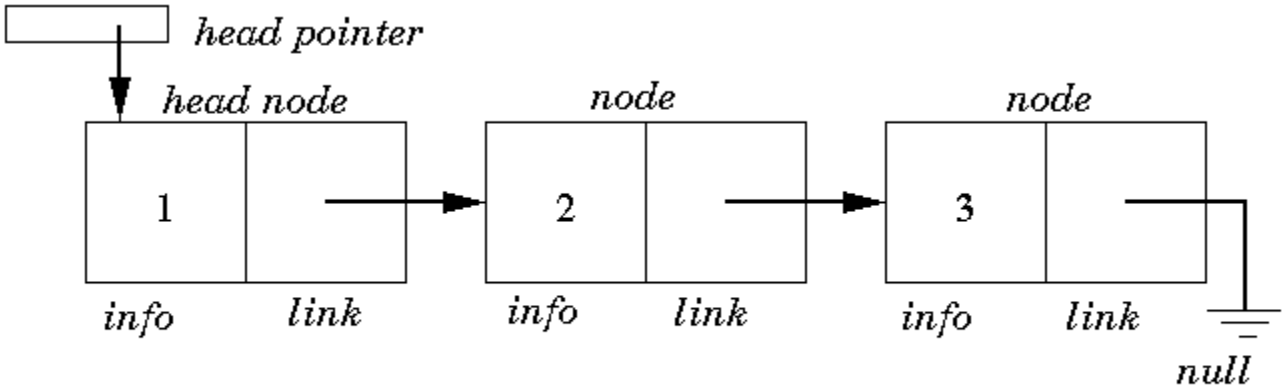


- A linked list, or one-way list is a linear collection of data elements called nodes, where linear order is given by means of pointer.

- A *linked list* is a series of connected *nodes*
- Each node contains at least
  - A piece of data (any type)
  - Pointer to the next node in the list
- *Head*: pointer to the first node
- The last node points to `NULL`

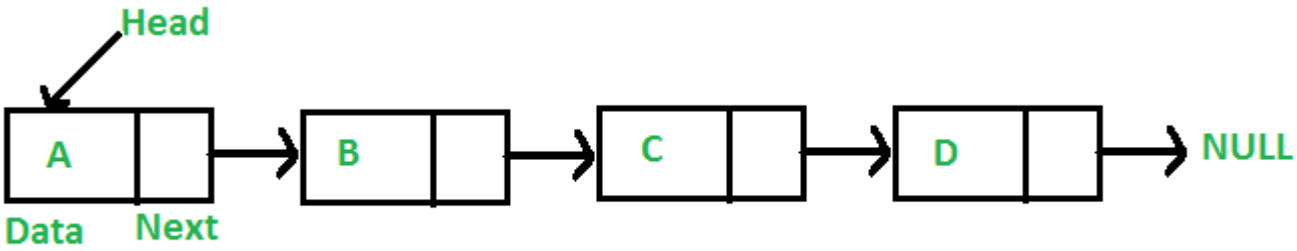


Representation of Linear Linked List:

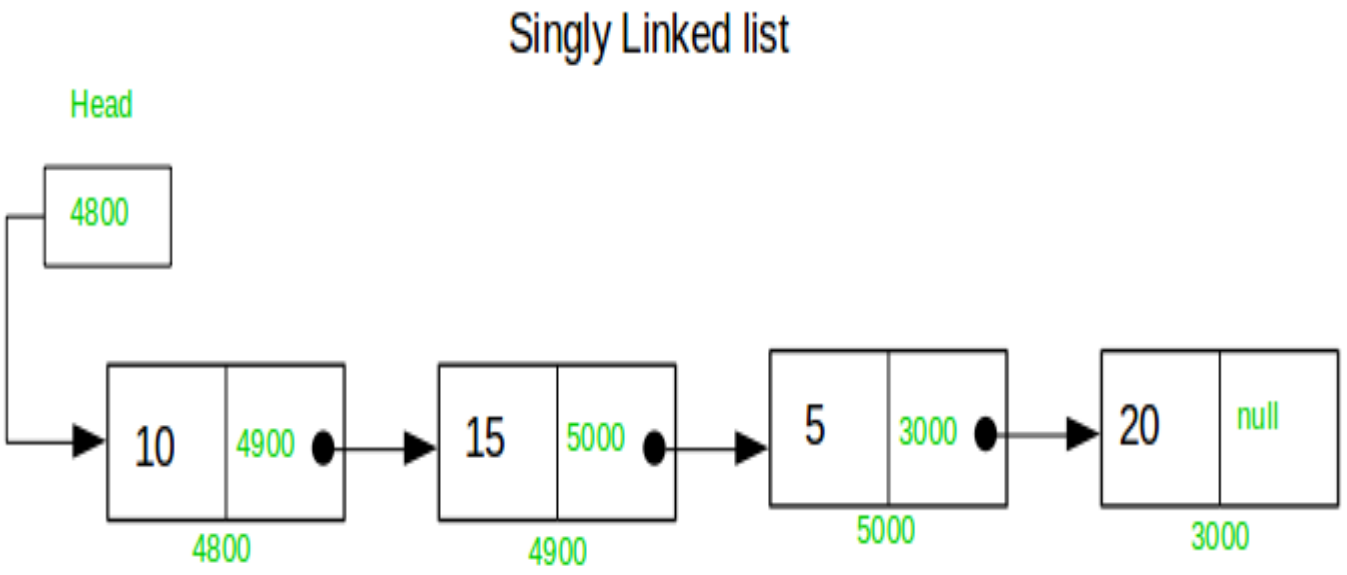


A Linked List

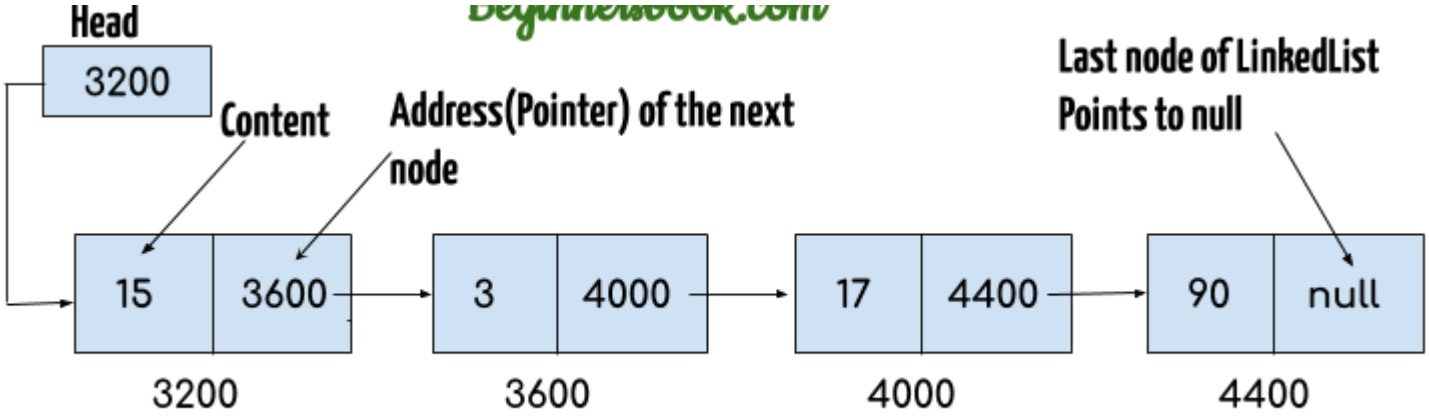
Example of linked list :



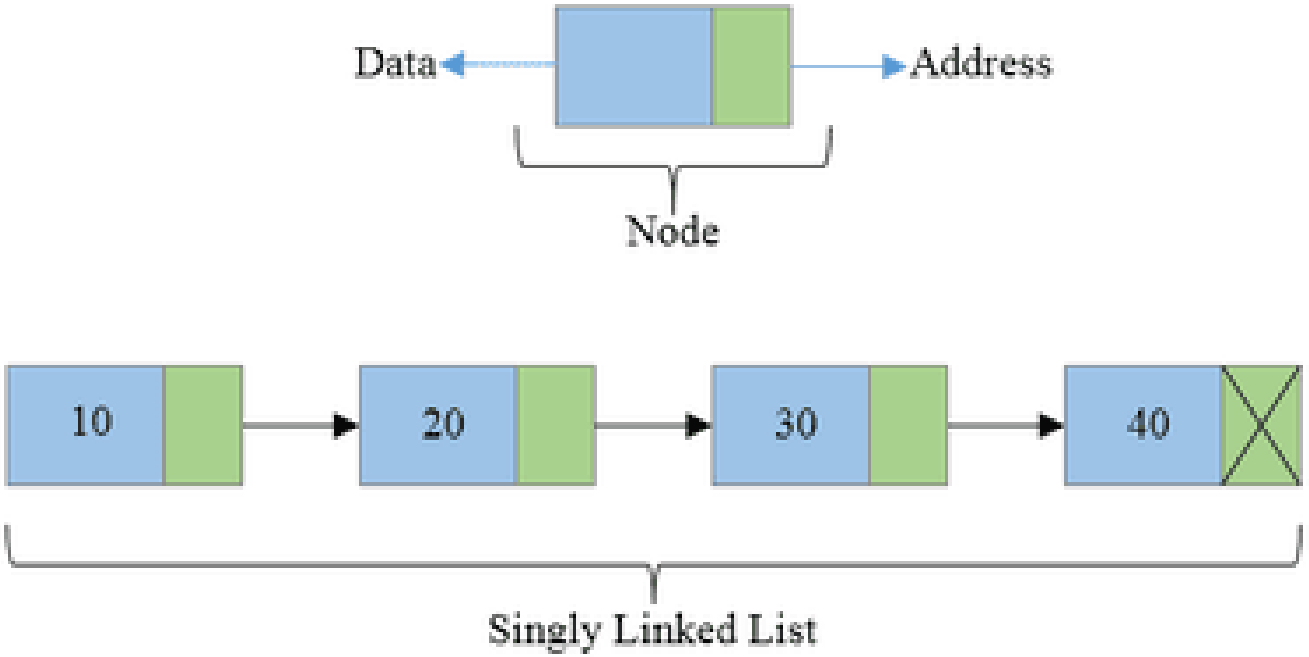
Example of Linked List:



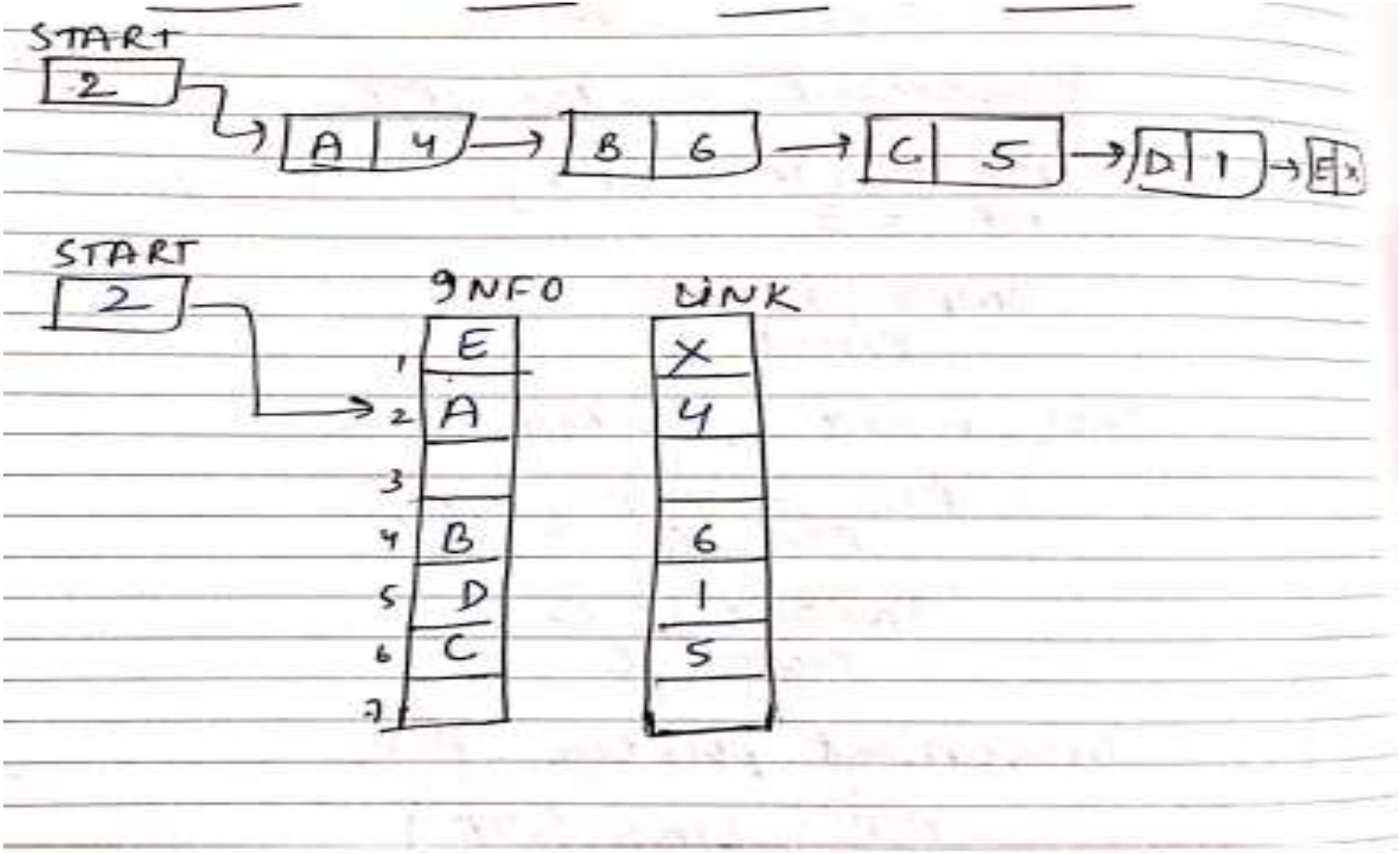
Example of linked list :



Example of LINKED List:



Representation in Memory:



PRACTICE QUESTION

START

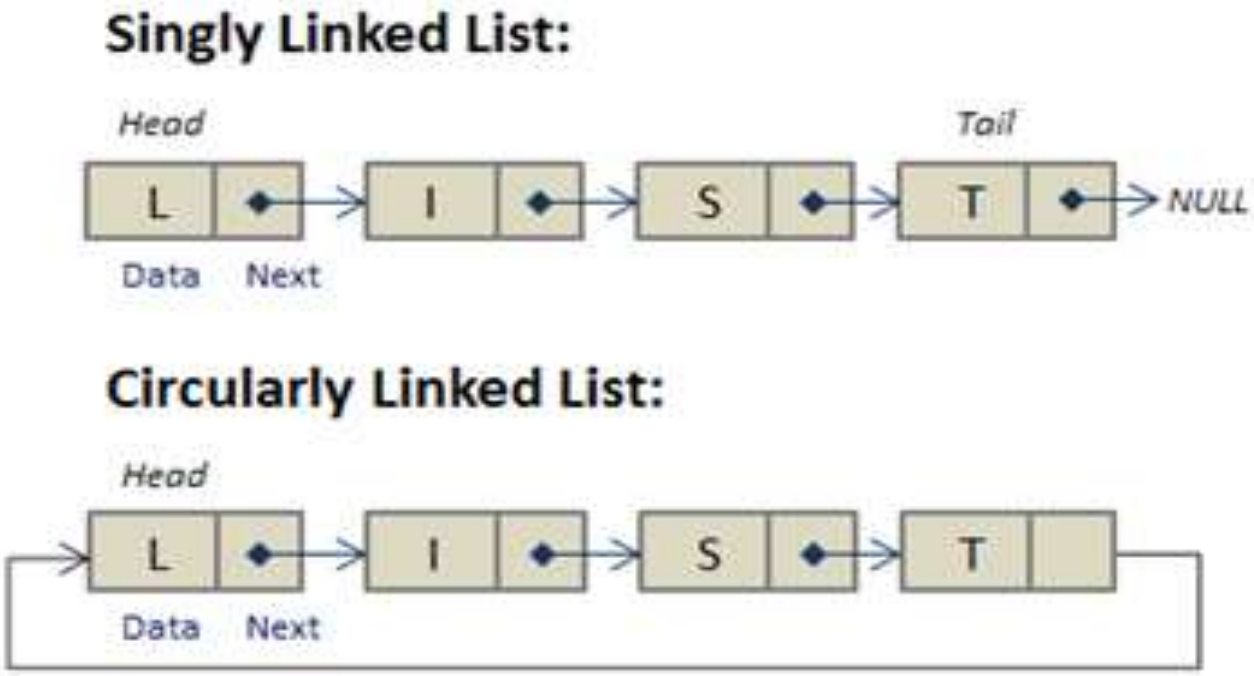
3

	INFO	LINK
1	A	5
2	A	4
3	P	1
4	K	X
5	L	2

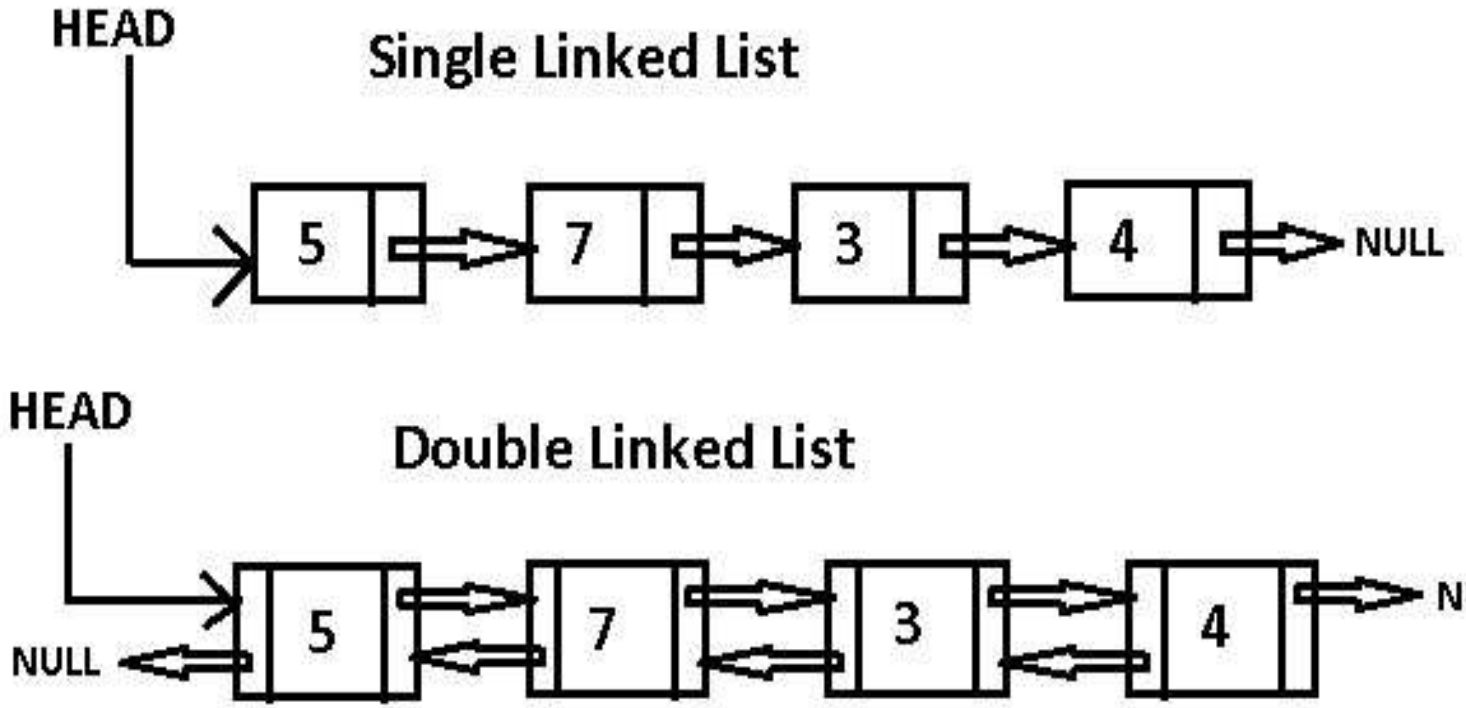
Types of Linked list:

- Linear or Single linked list
- Circular Linked List
- Two way or Doubly Linked list

Singly or Linear linked list
VS
Circular Linked List:



Single or Linear linked list VS Double Linked List:



Operations on Linear Linked List:
1. Traversing:

- **ALGORITHM:-**

TRAVERSE(LIST, START , PTR, INFO , LINK)

LIST is linear linked list stored in memory, START Pointer points to first node, PTR is pointer that points to the node currently being processed, INFO contains data elements and LINK that stores the address of next node.

1.Set PTR:=START. [Initializes pointer PTR]

2.Repeat steps 3 and 4 while PTR!=NULL.

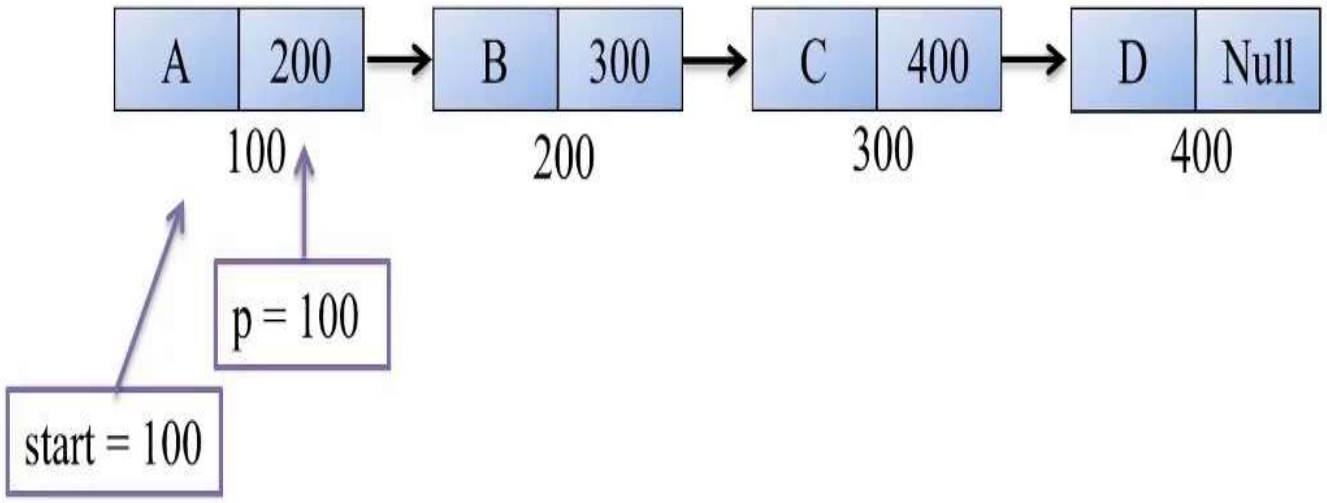
3.Apply PROCESS to INFO[PTR].

4.Set PTR:=LINK[PTR].[PTR now points the next node].

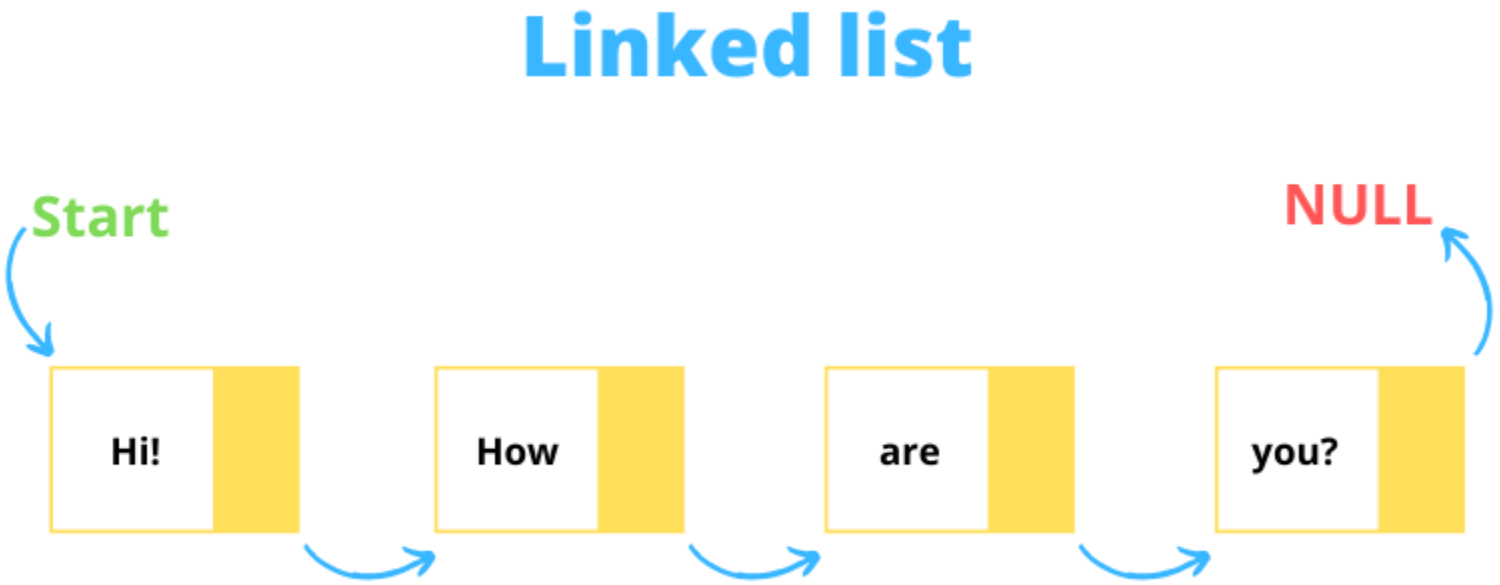
[End of step 2 loop]

5.Exit.

TRAVERSING A LINKED LIST EXAMPLE:



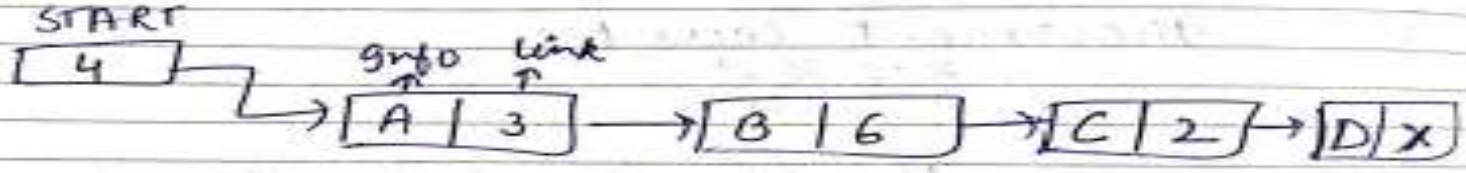
TRAVERSING A LINKED LIST EXAMPLE:



TRAVERSING SOLVED EXAMPLE 1:

① TRAVERSING accessing each element of a linked list exactly once.

Eg:- We want to print all the elements.



LIST be a linked list stored in memory
PTR is a pointer variable
that is initialized from START

START pointer stores the address of first data element of a linked list.

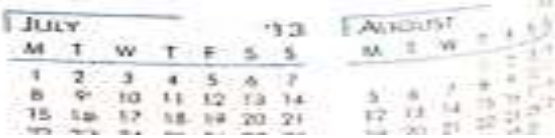
① Set PTR := START [Initialize PTR]

② while PTR ≠ NULL

③ Read all data elements or INFO part of each node.

④ PTR := 4

THINGS TO DO



TRAVERSING SOLVED EXAMPLE 1 contd:

THURSDAY

INFO[4] = A
Print A

(5) Increment pointer, PTR

PTR := LINK [PTR]
PTR := 3

INFO [3] = B
Print B

(6) Increment pointer, PTR

PTR := LINK [PTR]
PTR = 6

INFO [6] = C
Print C

(7) Increment pointer, PTR

PTR := LINK [PTR]
NOW, PTR = 2
INFO [2] = D
Print D

(8) Again, increment PTR

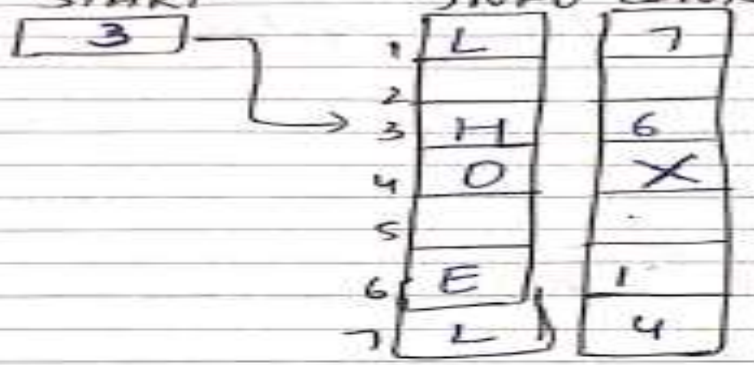
PTR := LINK [PTR]
NOW, PTR = NULL
Thus we exit from whi
loop!

THINGS TO DO

OCTOBER '13
M T W T F S S
1 2 3 4 5 6
7 8 9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31

TRAVERSING SOLVED EXAMPLE 2 :

TRAVERSING A LINKED LIST



Let us suppose we want to print all the elements of linked list.
Thus, we traverse the linked list.

(1) START = 3

(2) Initialize PTR with START
Thus set PTR := START
PTR := 3

Read the element present at 3rd memory location

THINGS TO DO (3) INFO [3] = H
Print H

SEPTEMBER '13
M T W T F S S
30 1
2 3 4 5 6 7 8

OCTOBER '13
M T W T F S S
1 2 3 4 5 6
7 8 9 10 11 12 13

TRAVERSING SOLVED EXAMPLE 2 contd:

(4) Increment pointer

PTR := LINK [PTR] [update pointer]

So PTR = 6

INFO [6] = E

Print E

(5) Increment pointer

PTR := LINK [PTR]

So PTR := 1

INFO [PTR] := INFO [1] = L

Print L

(6) Increment pointer

PTR := 7

INFO [7] := L

Print L

(7) Increment pointer

PTR := LINK [PTR]

PTR := 4

Thus, INFO [4] = 0

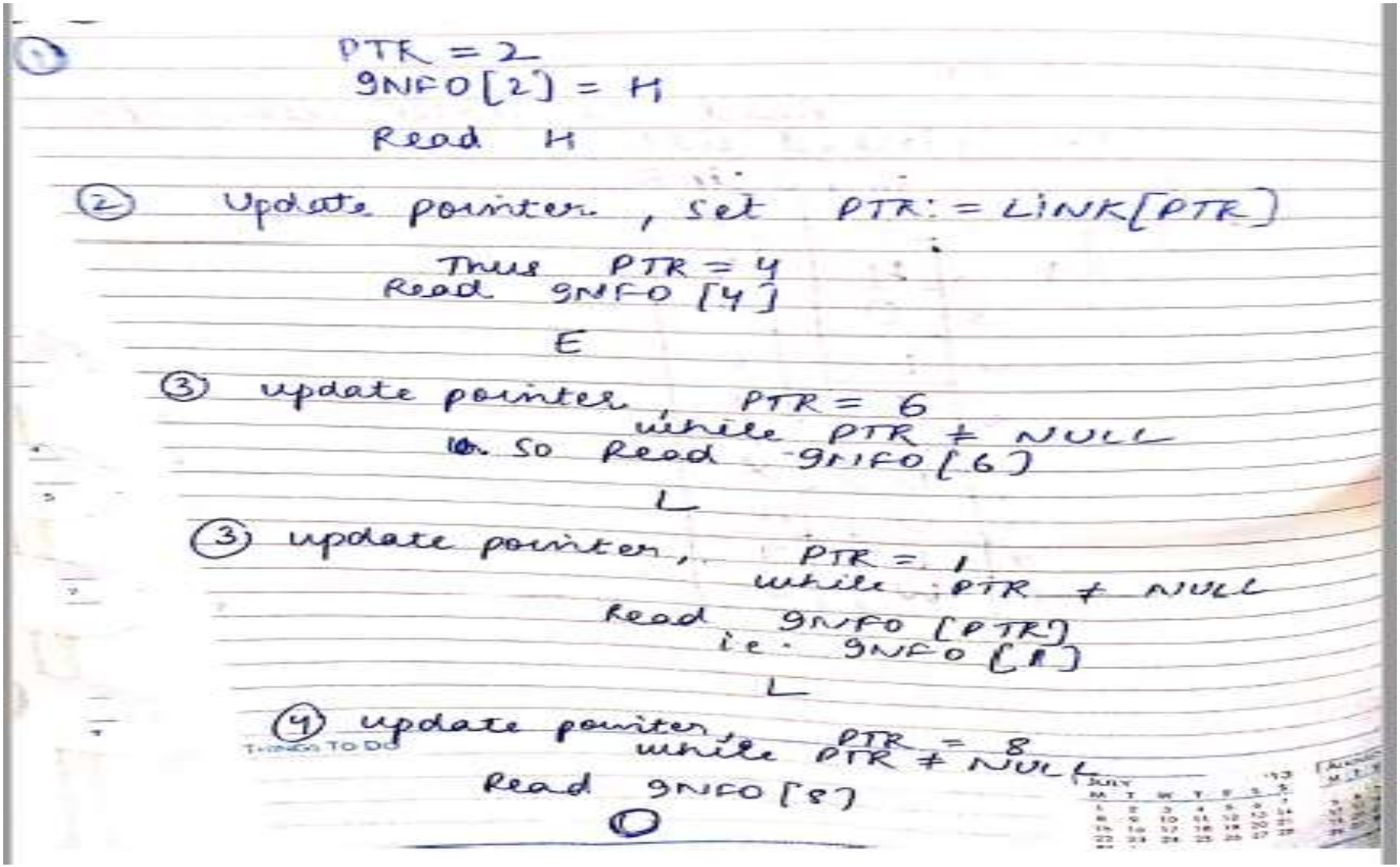
Print 0

TRAVERSING SOLVED EXAMPLE 3 :

START

	INFO	LINK
1	L	8
2	H	4
3	O	5
4	E	6
5	R	9
6	L	1
7	D	X
8	O	10
9	L	7
10	W	3

TRAVERSING SOLVED EXAMPLE 3 contd :



TRAVERSING SOLVED EXAMPLE 3 contd :

(5) update pointer, Set PTR := LINK [PTR]

Set PTR := 10

Read INFO[10]

W

(6) update pointer, Set PTR := 3

Read INFO [3] = O

(7) update pointer, Set PTR := 5

Read INFO[5]

R

(8) update pointer, Set PTR := 9

Read INFO[9]

L

(9) update pointer, Set PTR := 7

Read INFO[7]

D

Thus after traversing we get HELLOWORLD