Date:
Page:

DSA

Linear Array :-

LOC(LA[K]) = Base(LA) + w(K - lower bound)

w = word size of 1 element in memory
LA = Linear Array
K = value.

Two dimensional Array :-

Column major ⇒ Address(A[I][J]) = Base_Address + w{M(J-1) + I-1}

Row major ⇒ Address(A[I][J]) = Base_Address + w{N(I-1) + (J-1)}

w = no. of word stored in memory
m = no. of columns
n = no. of rows
I, J = subscript of the array element.

Date:
Page:

m x n.

Ques: consider 20x5 2 dimensional array marks which has its base address 1000 and the no. of words per memory location of the array is 2. Now compute the address of the element marks [18, 4] assuming that the elements are stored in row major order.

Row major ⇒

$$A[I][J] = Base\_Address + w\{N(I-1) + (J-1)\}$$

$$= 1000 + 2\{5(18-1) + (4-1)\}$$

$$2(20 \times 17 + 3)$$

1176

Two Dimensional Array

Column Major ⇒ $Address(A[I][J]) = B + w\{m(j-L1) + (i-L1)\}$

Row major ⇒ $Address(A[I][J]) = B + w\{n[i-L1] + [j-L2]\}$

No. rows

$U_1 - L_1 + 1$

$q_2 \cdot q_2 + 1$

$20 + 20 + 1$

$40 + 1$

Date:
Page:

Ques 2 Each element of an array a [-20..20, 10...35] requires 1 byte of storage if an array is column major implemented and the beginning of the array is at location 500. Determine the address of element a[0,30].

i j

column major

Address[A[I][J]] = B + w(m(j - L2) + (i - L1))

500 + 1(41(30 - 10) + (0 + 20))

⇒ 500 (820 + 20)

⇒ 500 + 840

⇒ 1340

20
41
20 41
2
82

1340
840
500

1340

840
2
3700

B = 5500

w = 1

Lowerbound of Row L_1 = -20

" " of column L_2 = 10

Upperbound of Row U_1 = 20

" " Column U_2 = 35

i = 0, j = 30

$m = U_1 - L_1 + 1$

$= 20 - (-20) + 1$

$= 41$

Putting values in formula

Add[0][30] = 5500 + 1(41(30 - 10) + 0 - (-20))

= 1340

✱ Calculate the address of X[4, 3] in a 2-dimensional array X[1...5, 1...4] stored in row major order in the main memory. Assume the base address to be 1000 and that element requires 4 words of storage.

Row major ⇒ Address (A[I][J]) = B + w{n[i-L1] + [j - L2]}

B = 1000

w = 4

n = $U_2 - L_2 + 1$

= 4 - 1 + 1

n = 4

i = 4

j = 3

X[4][3] = 1000 + 4{4[4-1] + [3-1]}

= 1000 + 56

= 1056 Ans

4 X 3

12 + 2

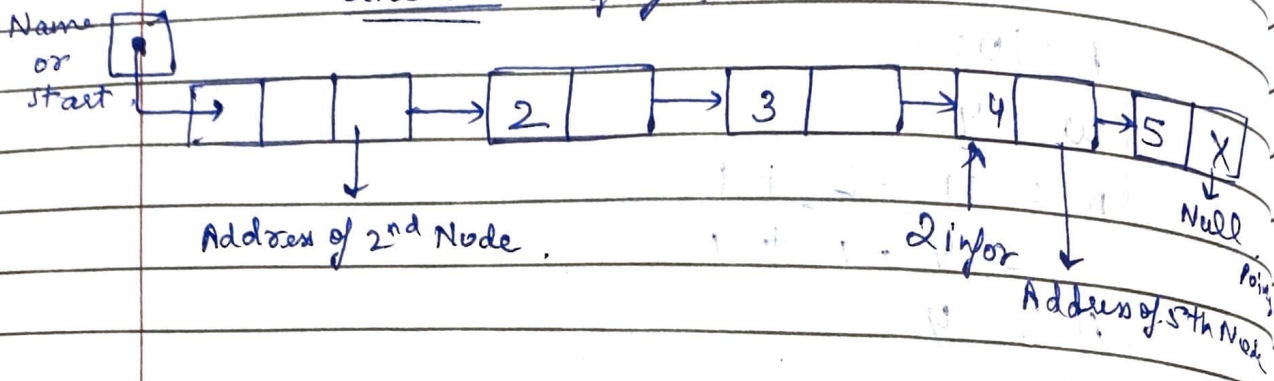
14

14

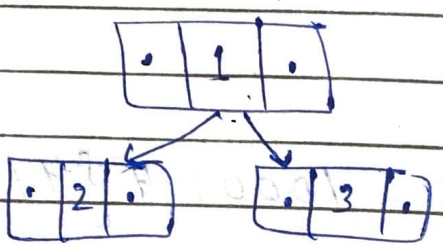
56

Linked List

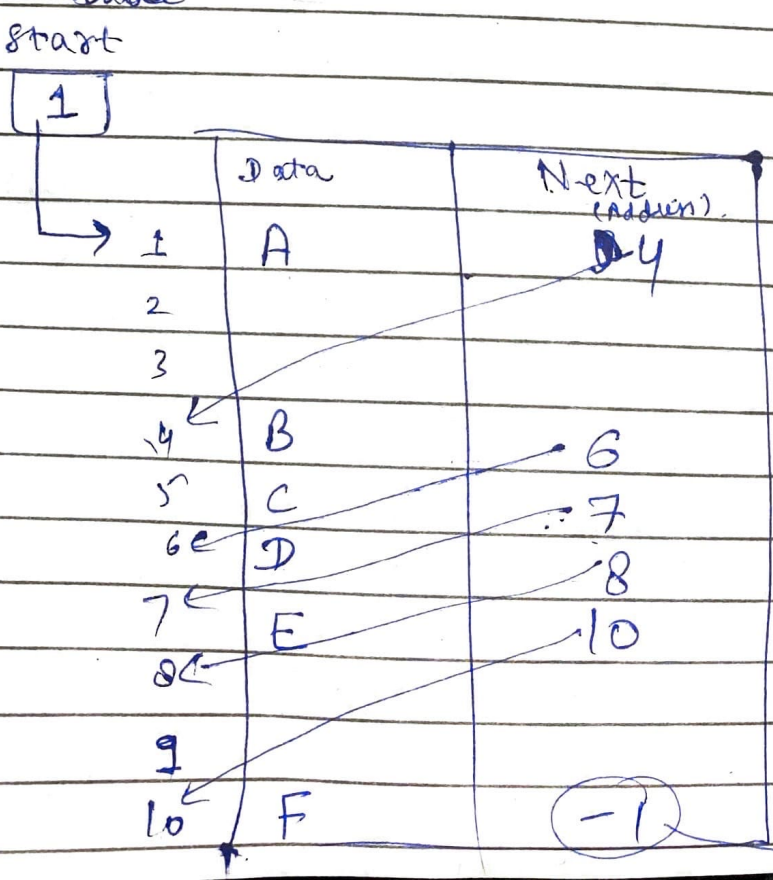
Linear → Singly



Non linear



Representation of Linked list in Memory



ALGORITHM

* Traversing of Linked List

TRAVESE (INFO, PIR, NEXT)

step 1: set PTR = START (Initialize pointer)
step 2: Repeat steps 3 and 4 while
PTR ≠ NULL
step 3: Apply Proces to INFO[PTR]
step 4: set PTR = PTR → NEXT
End of loop
step 5 : Exit

* Algorithm to print each node of LINKED LIST

PRINT (INFO, NEXT, START)

Step 1: Set PTR = START
Step 2: Repeat steps 3 and 4 while
PTR ≠ NULL
Step 3: Write : PTR → INFO or INFO [PTR]
Step 4: Set PTR = PTR → NEXT
End of Loop
Step 5: Exit

Date:
Page:

* Algorithm to print the no. of nodes in the Linked List

COUNT(INFO, PTR, N, START)

Step 1: Set N = 0
Step 2: Set PTR = START
Step 3: Repeat step 4 and 5 while PTR ≠ NULL
Step 4: Print; N = N + 1
Step 5: set PTR = PTR → NEXT
End of loop
Step 6: exit

DSA

Deleting a Node from LL.

1. The first Node is deleted
2. The last Node is deleted
3. The node after a given node is deleted
4. The node is deleted from a sorted LL.

(1) The first Node is deleted :-

Algo :-

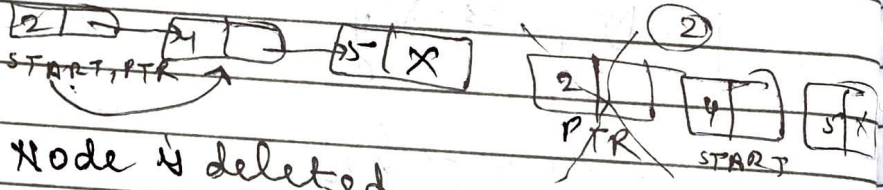
Step 1: If START = NULL, then
Write UNDERFLOW
Go to step 5 [End of if]

Step 2: Set PTR = START

Step 3: Set START = START → NEXT

Step 4: FREE PTR

Step 5: EXIT



(2) The last Node is deleted

Step 1: If START = NULL, then write UNDERFLOW
Go to step 8 [End of if]

Step 2: Set PTR = START

Step 3: Repeat step 4 and 5 while
PTR → NEXT != NULL

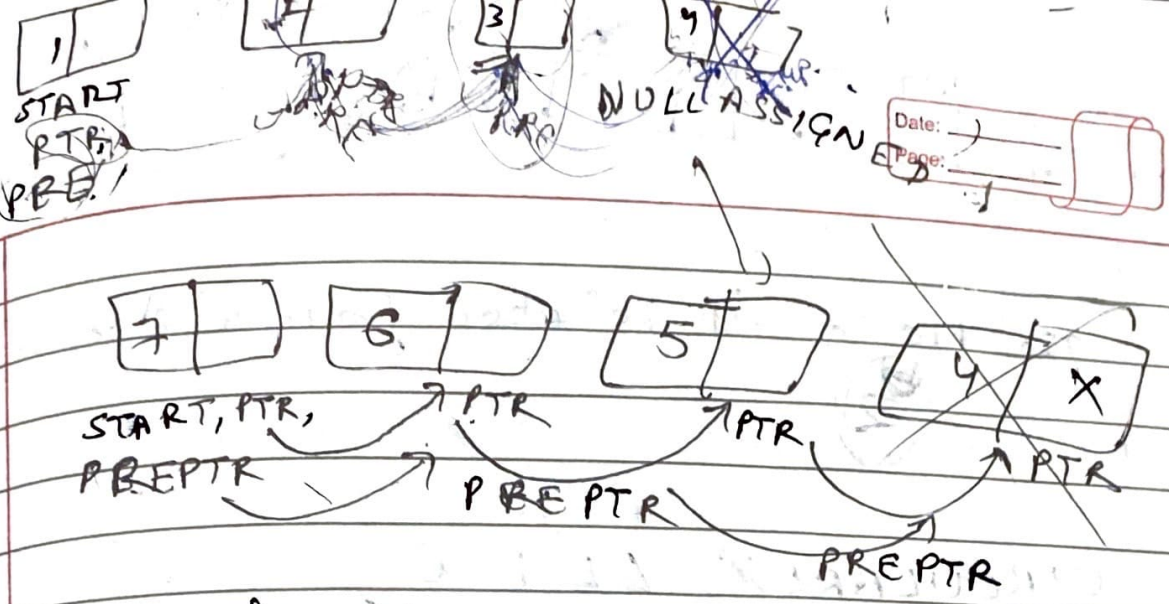
Step 4: Set PREPTR = PTR

Step 5: Set PTR = PTR → NEXT
[End of loop]

Step 6: Set PREPTR → NEXT = NULL

Step 7: FREE PTR

Step 8: Exit.



(3) The node after a given node is deleted

Step 1: If START = NULL, then write UNDERFLOW
Go to step 10 [End of if]

Step 2: Set PTR = START

Step 3: Set PREPTR = PTR

Step 4: Repeat step 5 and 6 while
PREPTR → DATA ! = NUM

Step 5: Set PREPTR = PTR

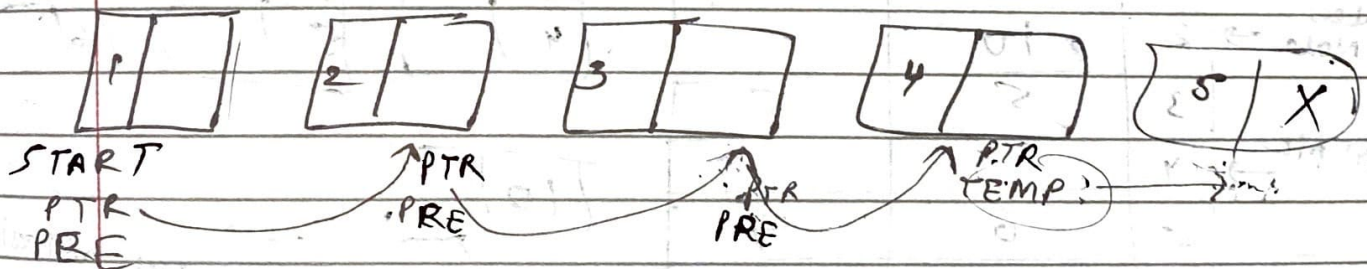
Step 6: Set PTR = PTR → NEXT
[End of loop]

Step 7: Set TEMP = PTR → NEXT

Step 8: Set PREPTR → NEXT = TEMP → NEXT

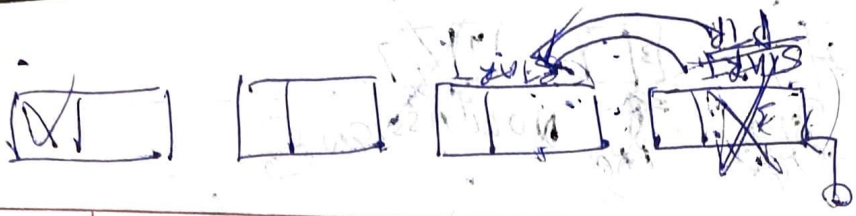
Step 9: FREE TEMP

Step 10: Exit.



PREPTR → NEXT = TEMP → NEXT

Temp ke next me pre PTR ke next ka address store kia.



Date:
Page:

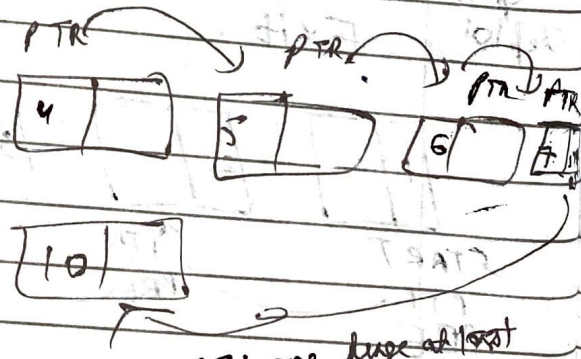
4. The Node is deleted from a sorted LL

CIRCULAR LL : Beg.

INSERTION

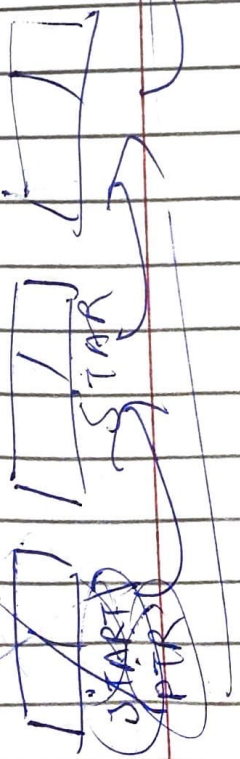
Step 1: If AVAIL = NULL, then write OVERFLOW
Go to Step 11 (End of if)
Step 2: Set New-Node = AVAIL
Step 3: Set AVAIL = AVAIL → NEXT
Step 4: Set New_Node → DATA = VAL
Step 5: Set PTR = START
Step 6: Repeat step 7 while PTR → NEXT != START
Step 7: PTR = PTR → NEXT
Step 8: Set New_Node → Next = START
Step 9: Set PTR → NEXT = New_Node
Step 10: set START = New_Node
Step 11: EXIT.

		Data	Next
PTR START →	1	4	3
New Node →	2	10	
	3	5	5
AVAIL →	4		
	5	6	7
	6		
	7	7	1
	8		
	9		



INSERTION after given node in LL:

step 1: If AVAIL = Null, then write OVERFLOW
Go to step 12 (End of if)
step 2: set NEW_NODE = AVAIL
step 3: set AVAIL = AVAIL → NEXT
step 4: Set New_Node → Data = VAL
step 5: set PTR = START
step 6: set PREPTR = PTR
step 7: Repeat step 8 and 9 while PREPTR → DATA != NUM
step 8: set PREPTR = PTR
step 9: set PTR = PTR → NEXT
[End of loop]
step 10: PREPTR → NEXT = New_Node
step 11: set New_Node → NEXT = PTR
step 12: EXIT



[redacted]

Date:
Page:

Insertion at end in Circular LL.

Step 1: If AVAIL = NULL, then write OVERFLOW
Go to step 10, [End of if]

Step 2: set New_Node = AVAIL

Step 3: Set AVAIL = AVAIL → NEXT

Step 4: Set New_Node → DATA = VAL

Step 5: Set PTR = START

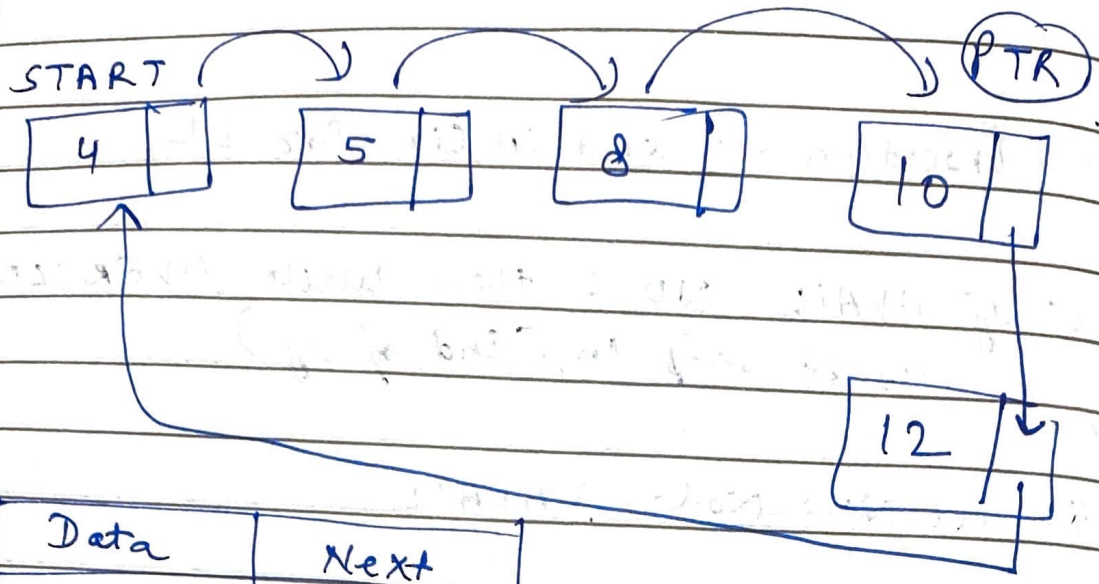
Step 6: Repeat step 7 while PTR → NEXT != START

Step 7: Set PTR = PTR → Next

Step 8: Set PTR → NEXT = New_Node

Step 9: Set New_Node → NEXT = START

Step 10: Exit.



START		Data	Next
	1	4	3
New_Node →	2	12	
	3	5	4
	4	8	6
AVAIL →	5		
	6	10	1
	7		
	8		

Deleting at Beginning of Circular LL:

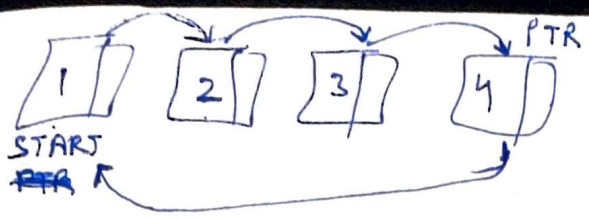
Step 1: If START = NULL, then write UNDERFLOW
Go to step 8, (end of if)

Step 2: Set PTR = START

Step 3: Repeat Step 4 while PTR → NEXT != START

Step 4: Set PTR = PTR → NEXT

Step 5: Set PTR → NEXT = START → NEXT

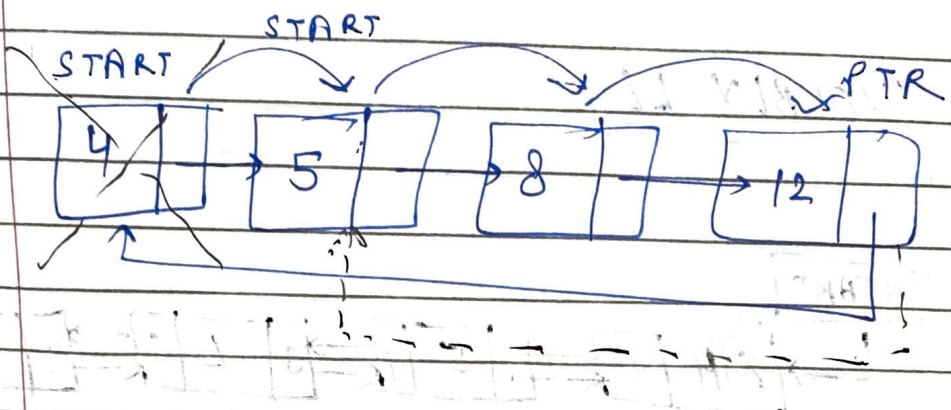


Date:
Page:

Step 6: FREE START

Step 7: Set START = PTR → NEXT

Step 8: Exit



Deleting at End of Circular LL.

Step 1: If START = NULL, then write UNDERFLOW
Go to Step 8, (End of if).

Step 2: Set PTR = START

Step 3: Repeat Step 4 & 5 while PTR → NEXT != START

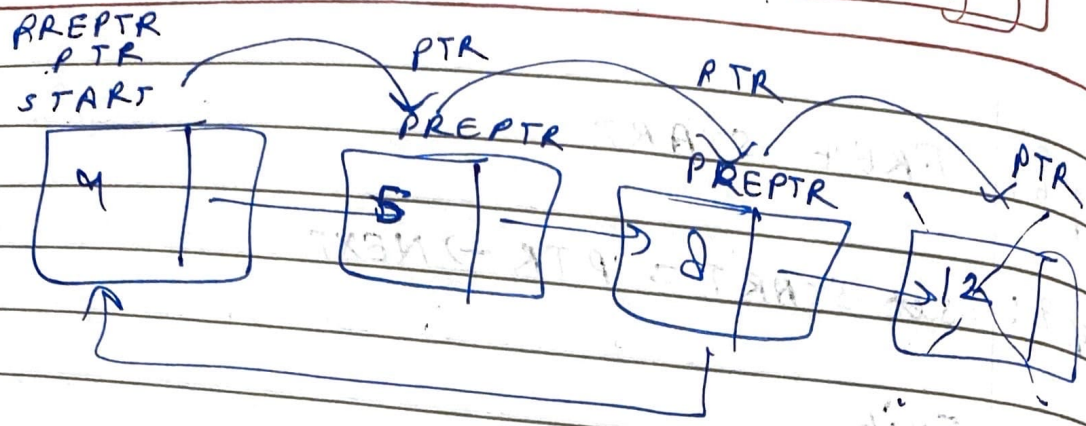
Step 4: set PREPTR = PTR

Step 5: Set PTR = PTR → NEXT

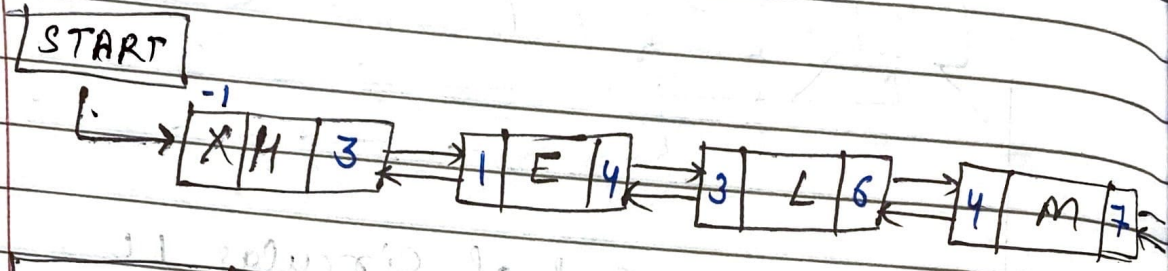
Step 6: set PREPTR → NEXT = START

Step 7: FREE PTR

Step 8: Exit.

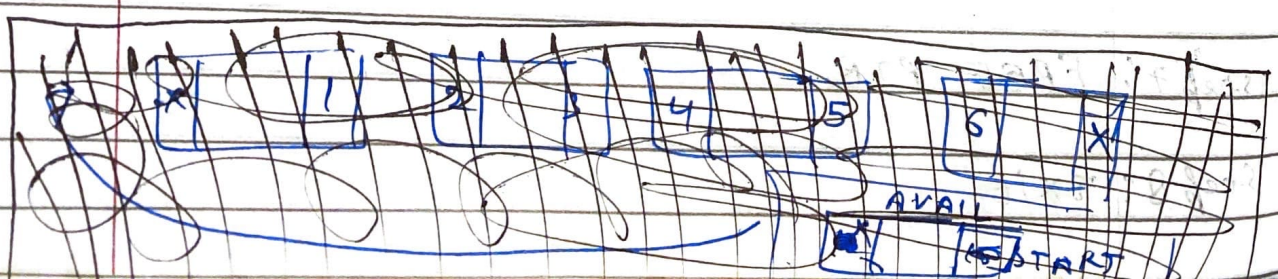


DOUBLY LL



	DATA	PREV	NEXT
START 1	H	-1 NULL	3
2			
3	E	1	4
4	L	3	6
5			
6	M	4	7
7		6	-1 NULL
8			

Ya do start ka previous NULL hoga ye to last ka next NULL hoga.



Date:
Page:

Insertion Doubly LL at Beginning :-

step 1: If AVAIL = NULL, then write OVERFLOW
Go to step 8 [End of if]

step 2: set New_Node = AVAIL

step 3: set AVAIL = AVAIL → NEXT

step 4: Set New_Node → Data = VAL

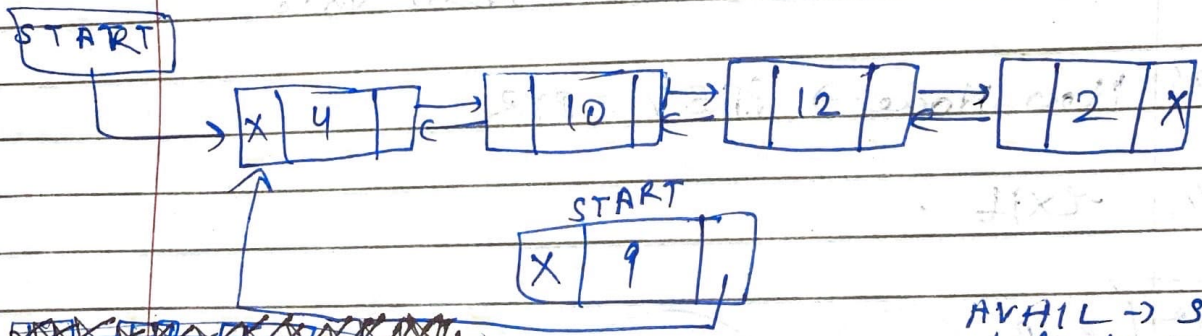
6	0	X

step 5: set New_Node → PREV = NULL

step 6: Set New_Node → NEXT = START

step 7: set START = New_Node

step 8: exit



		Data	PREV	NEXT
START	1			
	2			
New Node	3			
	4			
AVAIL	5			
	6			

AVAIL → Sabse pehle khali data ko AVAIL BOLTE HAI ⇒

Date:
Page:

Insertion Double LL at End.

Step 1: If AVAIL = NULL, then write OVERFLOW
Go to step 11 [End of it].

Step 2: set New_Node = AVAIL

Step 3: set AVAIL = AVAIL → NEXT

Step 4: Set New_Node → Data = VAL

Step 5: Set New_Node → NEXT = NULL

Step 6: set PTR = START

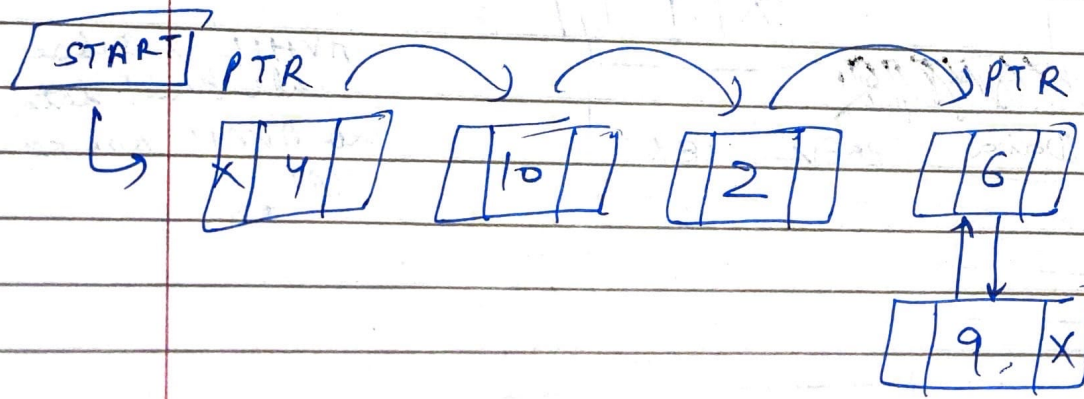
Step 7: Repeat step 8 while PTR → NEXT != NULL

Step 8: Set PTR = PTR → NEXT

Step 9: Set PTR → NEXT = New_Node

Step 10: New_Node → PREV = PTR

Step 11: Exit.



Date:
Page:

Doubly LL → Deletion in Beginning.

Step 1: If START = NULL, then write UNDERFLOW
Go to step 6 (End of it)

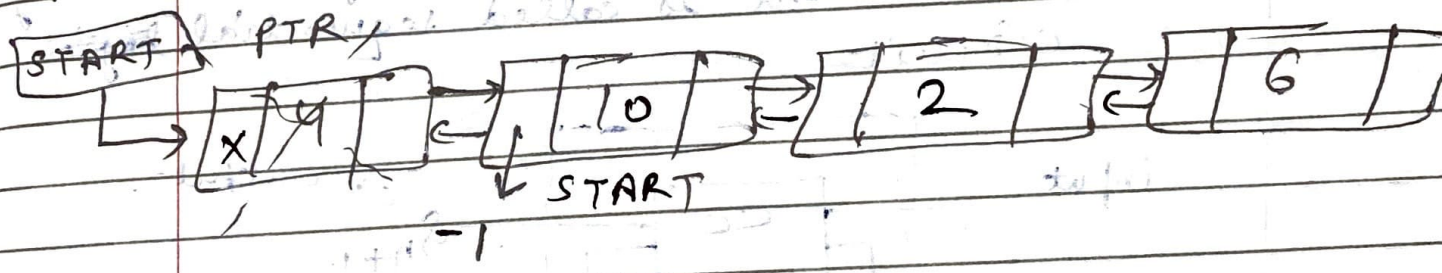
Step 2: set PTR = START

Step 3: Set START = START → NEXT

Step 4: Set START → PREV → NULL

Step 5: FREE PTR

Step 6: Exit



Double LL → Deletion at Last.

Step 1: If START = NULL, then write UNDERFLOW
Go to step 7 (End of it)

Step 2: Set PTR = START

Step 3: Repeat Step 4 while PTR → NEXT != NULL

Step 4: Set PTR = PTR → NEXT

Step 5: Set PTR → PRE → NEXT = NULL.

Step 6: FREE PTR

Step 7: Exit.