☺ DOUBLY LINKED LIST ☺

START

```
→ X  1 | ⇄ | 2 | ⇄ | 5  X
```

A doubly linked list or a two way linked list is a more complex type of linked list which contains a pointer to the next as well as the previous node in the sequence. Therefore it consists of Three parts → data, a pointer to the next node, and pointer to the previous node.

In C, structure of a doubly linked list can be given as:-

```
        struct node
        {
              struct node *prev;
              int data;
              struct node *next;
        };
```

PREV → Store address of the previous node. help use in backward traversing.

NEXT → Store address of the next node.

The PREV field of the first node and the NEXT field of the last node will contain NULL.

On seeing that a doubly linked list calls for more space per node and more expensive basic operations. The main advantages of using a doubly linked list is that it makes searching twice as efficient.

START | 1 |

	DATA	PREV	NEXT
1	H	-1	3
2			
3	E	1	6
4			
5			
6	L	3	7
7	L	6	9
8			
9	O	7	-1

INSERTING A NEW NODE IN A DOUBLY LINKED LIST

(a) ☺ Inserting a Node at the Beginning of a Doubly Linked List

| X | 1 | | | 7 | | | 3 | | | 4 | | | 2 | X |

START

Allocate memory for the new node and initialize its DATA part to 9 and PREV field to NULL.

| X | 9 | |

Add the new node before the START Node. Now the new node becomes the first node of the list.

| X | 9 | | | 1 | | | 7 | | | 3 | | | 4 | | 2 | X |

ALGORITHMS

Steps

1 : IF AVAIL = NULL

Write Overflow

Go to Step 9

[END OF IF]

2: SET NEW-NODE = AVAIL

3: SET AVAIL = AVAIL → NEXT

4: SET NEWNODE → DATA = VAL

5: SET NEWNODE → PREV = NULL

6: SET NEWNODE → NEXT = START

7: SET START → PREV = NEW-NODE

8. SET START = NEWNODE

9: EXIT.

(b) Inserting a Node at the End of a Doubly Linked List

Step 1: IF AVAIL = NULL
    Write OVERFLOW
    GoTo 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 NEW-NODE → NEXT = NULL
Step 6: SET PTR = START
Step 7: Repeat Step 8 while PTR → NEXT != NULL
Step 8:   SET PTR = PTR → NEXT
    [END OF LOOP]
Step 9: SET PTR → NEXT = NEW-NODE
Step 10: SET NEW-NODE → PREV = PTR
Step 11: EXIT

START

Allocate memory for the new node and initialize its DATA part to 9 bits
NEXT field to NULL.

Take a pointer variable PTR and make its point to the first node of the list.

START, PTR

move PTR so that it points to the last node of the list. Add new node after the node pointed by PTR.

START   PTR

© ⓘ Inserting a new node after a given node in a Doubly linked list.

Step1: IF AVAIL = NULL

Write OVERFLOW

GOTO STEP 12

[END OF IF]

Step2: SET NEW-NODE = AVAIL

Step3: SET AVAIL = AVAIL → NEXT

Step4: SET NEW-NODE → DATA = VAL

Step5: SET PTR = START

Step6: Repeat Step7 while PTR → DATA != NUM

Step7: SET PTR = PTR → NEXT

[END OF LOOP]

Step8: SET NEW NODE → NEXT = PTR → NEXT

Step9: SET NEW NODE → PREV = PTR

Step10: SET PTR → NEXT = NEW-NODE

Step11: SET PTR → NEXT → PREV = NEW NODE

Step12: EXIT

Insert node (new node) blw PTR and the node succeeding it.

☺ Inserting a node before a given Node in a Doubly linkedlist

Step1: IF AVAIL = NULL
     Write Overflow
     Goto Step 12.
     [END OF IF]

Step2: SET NEW-NODE = AVAIL

Step3: SET AVAIL = AVAIL → NEXT

Step4: SET NEW-NODE → DATA = VAL

Step5: SET PTR = START

Step6: Repeat Step7 While PTR → DATA ! = NUM

Step7: SET PTR = PTR → NEXT
     [END OF LOOP]

Step8: SET NEW-NODE → NEXT = PTR

Step9: SET NEW-NODE → PREV = PTR → PREV

Step10: SET PTR → PREV = NEWNODE

Step11: SET PTR → PREV → NEXT = NEW-NODE

Step12: EXIT

Add the new node in between the node pointed by PTR and the node preceding it.

☺ Deleting a Node from a Doubly Linked List

(a) Delete the first Node :→

Step 1: IF START = NULL

         Write UNDERFLOW

         GO TO STEP 6

         [END OF IF]

Step 2: SET PTR = START

Step 3: SET START = START → NEXT

Step 4: SET START → PREV = NULL

Step 5: FREE PTR

Step 6: EXIT.

START

Free the memory occupied by the first node of the list and make the second node of the list as the START NODE.

START

(b) Deleting the last node from a Doubly linked list

Step 1: IF START = NULL

         Write UNDERFLOW

         GO TO STEP 7

         [END OF IF]

Step 2: SET PTR = START

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

Step 4: SET PTR = PTR → NEXT

Step 5: SET PTR → PREV → NEXT = NULL

Step 6: FREE PTR

Step 7: EXIT.

```
| X | 1 | ⇄ | 3 | ⇄ | 5 | ⇄ | 7 | ⇄ | 8 | ⇄ | 9 | X |
```

START , PTR

Take a pointer variable PTR that points to the first node of the list.

Move PTR to the last node of the list.

```
| X | 1 | ⇄ | 3 | ⇄ | 5 | ⇄ | 7 | ⇄ | 8 | X |
```

START.

(c) (ii) Deleting the Node After a Given Node in a Linked List

Step 1: IF START = NULL

        write Underflow

        GoTo STEP 9

       [END OF IF]

Step 2: SET PTR = START

Step 3: Repeat Step 4 While PTR→DATA != NUM

Step 4:   SET PTR = PTR→NEXT

       [END OF LOOP]

Step 5: SET TEMP = PTR→NEXT

Step 6: SET PTR→NEXT = TEMP→NEXT

Step 7: SET TEMP→NEXT→PREV = PTR

Step 8: FREE TEMP

Step 9: EXIT

Deleting the Node Before a given Node:-

Step 1: IF START = NULL

Write UNDERFLOW

GOTO STEP 9

[END OF IF]

Step 2: SET PTR = START

Step 3: Repeat Step 4 while PTR → DATA! = NUM

Step 4: SET PTR = PTR → NEXT

[END OF LOOP]

Step 5: SET TEMP = PTR → PREV

Step 6: SET TEMP → PREV → NEXT = PTR

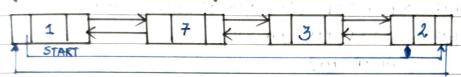Step 7: SET PTR → PREV = TEMP → PREV

Step 8: FREE TEMP

Step 9: EXIT.

CIRCULAR DOUBLY LINKED LIST

Date : _____

The difference b/w a Doubly linked list and Circular linked list → Circular Doubly linked list having No NULL value either at first node or the last nodes. First Node previous contain address of the last node and Next of last node contain address of the first node.

Main advantage of using a Circular Doubly linked list is that it make a Searching twice as efficient.

INSERTING A NEW NODE IN C.D.L.L

(a). Inserting a node at the Beginning.


```
1  |  7  |  3  |  2
START
```

Allocate memory for the new node and Initialize NewNode with DATA 9.

Step 1: IF AVAIL= NULL

        Write OVERFLOW

          Go To STEP 13

       [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

       [END OF LOOP]

Step 8: SET PTR → NEXT = NEWNODE

Step 9: SET NEWNODE → PREV = PTR
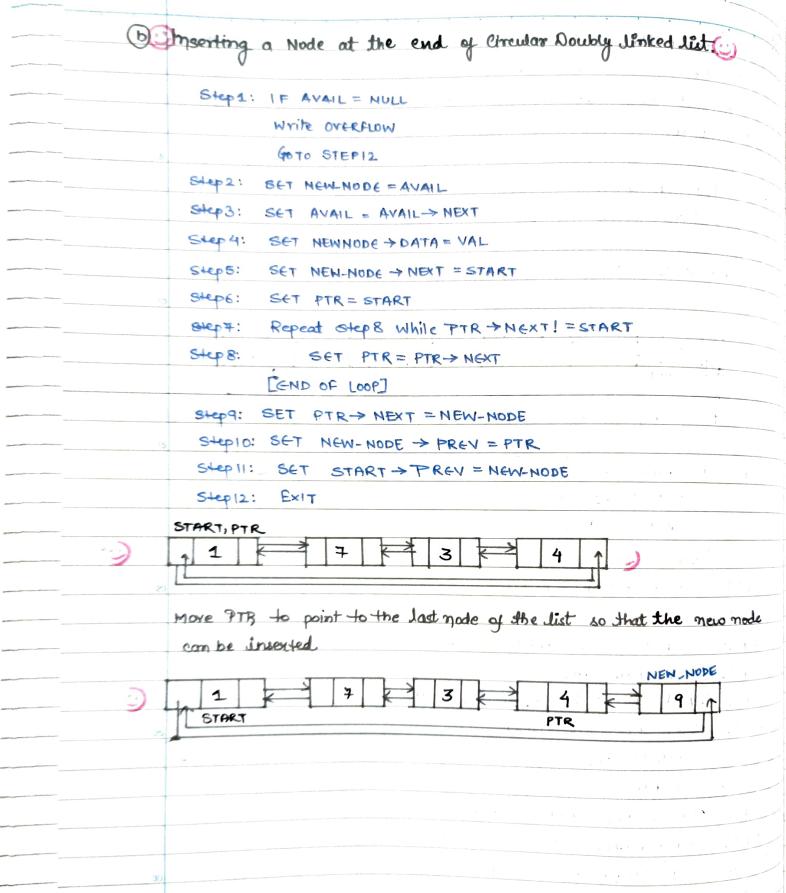
Step 10: SET NEWNODE → NEXT = START

Step 11: SET START → PREV = NEW-NODE

Step 12: SET START = NEW_NODE

Step 13: EXIT,

(b) Inserting a Node at the end of Circular Doubly linked list.

Step1: IF AVAIL = NULL

Write OVERFLOW

GOTO STEP12

Step2: SET NEW-NODE = AVAIL

Step3: SET AVAIL = AVAIL → NEXT

Step4: SET NEWNODE → DATA = VAL

Step5: SET NEW-NODE → NEXT = START

Step6: SET PTR = START

Step7: Repeat Step8 while PTR → NEXT! = START

Step8: SET PTR = PTR → NEXT

[END OF LOOP]

Step9: SET PTR → NEXT = NEW-NODE

Step10: SET NEW-NODE → PREV = PTR

Step11: SET START → PREV = NEW-NODE

Step12: EXIT

START, PTR

Move PTR to point to the last node of the list so that the new node can be inserted.

CIRCULAR
DELETING NODE FROM THE DOUBLY LINKED LIST

(a). Deleting the node (FIRST NODE) from the Circular Doubly Linked List. ☺

Step 1:   IF START = NULL

           Write UNDERFLOW

           GOTO 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

           [END OF LOOP]

Step 5:   SET PTR → NEXT = START → NEXT

Step 6:   SET START → NEXT → PREV = PTR

Step 7:   FREE START

Step 8:   SET START = PTR → NEXT.

(b) Deleting the last Node from the Circular Doubly Linked List ☺

Step 1:   IF START = NULL

           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

           [END OF LOOP]

Step 5:   SET PTR → PREV → NEXT = START

Step 6:   SET START → PREV = PTR → PREV

Step 7:   FREE PTR

Step 8:   EXIT