## DOUBLY LINKED LIST

A doubly linked list is one in which all nodes are linked together by multiple links which help in accessing both the successor (next) and predecessor (previous) node for any arbitrary node within the list. Every nodes in the doubly linked list has three fields: LeftPointer, RightPointer and DATA. Fig. 5.22 shows a typical doubly linked list.

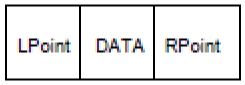

**Fig. 5.24.** A typical doubly linked list node

LPoint will point to the node in the left side (or previous node) that is LPoint will hold the address of the previous node. RPoint will point to the node in the right side (or nex node) that is RPoint will hold the address of the next node. DATA will store the information of the node.

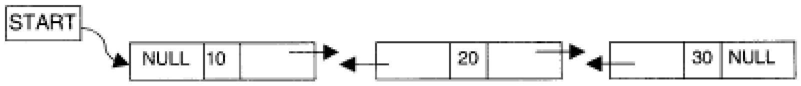

**Fig. 5.25.** Doubly Linked List

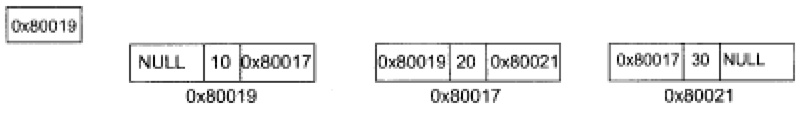

**Fig. 5.26.** Memory Representation of Doubly Linked List

Representation of doubly linked list

A node in the doubly linked list can be represented in memory with the following declarations.

```
Struct Node
{
    Int DATA;
    Node *next;
    Node *prev;
};
```

All the operations performed on singly linked list can also be performed on doubly linked list. Following figure will illustrate the insertion and deletion of nodes.

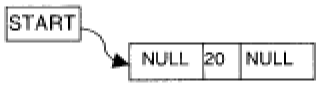

**Fig. 5.27.** Add(20)

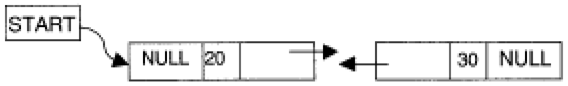

**Fig 5.28.** Insert (30) at the end

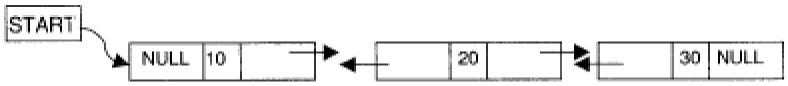

**Fig 5.29.** Insert (10) at the beginning

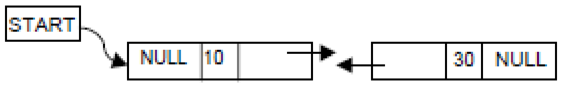

**Fig 5.30.** Delete a node at the 2nd position

**Algorithm for creating a doubly linked list (inserting at the end)**

```
1. Input the DATA to be inserted
2. TEMP to create a new node
3. TEMP->info=DATA
4. TEMP->next=NULL
5. if (START is equal to NULL)
    5.a. TEMP->prev=NULL
    5.b. START=TEMP
    5.c. exit
6. else
    6.a. HOLD =START
    6.b. while(HOLD->next not equal to NULL)
        6.b.1 HOLD=HOLD->next
    6.c. HOLD->next=TEMP
    6.d. TEMP->prev=HOLD
7. exit
```

**Algorithm for inserting a node at the beginning**

```
1. Input the DATA to be inserted
2. TEMP to create a new node
3. TEMP->prev=NULL
4. TEMP->info=DATA
5. TEMP->next=START
6. if (START is equal to NUUl)
    6.a. START=TEMP
    6.b. exit
7. START->prev=TEMP
8. START=TEMP
9. exit
```

**Algorithm for inserting a node at a specific position**

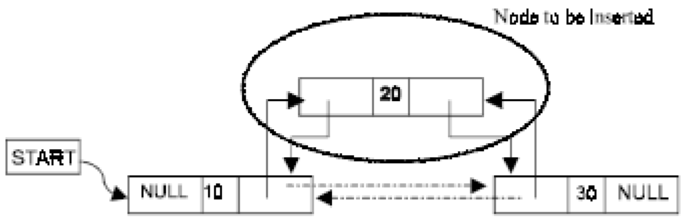

**Fig. 5.31.** Insert a node at the 2nd position

Suppose START is the first position in linked list. Let DATA be the element to be inserted in the new node. POS is the position where the NewNode is to be inserted. TEMP is a temporary pointer to hold the node address.

```
1. Input the DATA and POS
2. Initialize TEMP = START; i = 0
```

```
3. Repeat the step 4 if (i less than POS) and (TEMP is not equal to NULL)
4. TEMP = TEMP ->next; i = i +1
5. If (TEMP not equal to NULL) and (i equal to POS)
    (a) Create a New Node
    (b) NewNode -> DATA = DATA
    (c) NewNode -> next = TEMP -> next
    (d) NewNode -> prev = TEMP
    (e) (TEMP -> next) -> prev = NewNode
    (f ) TEMP -> next = New Node
6. Else
    (a) Display "Position NOT found"
7. Exit
```

**Algorithm for deleting a node**

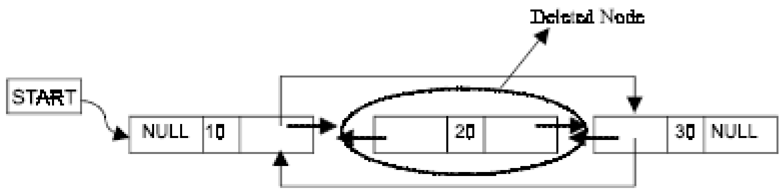

Fig. 5.32. Delete a node at the 2nd position

Suppose START is the address of the first node in the linked list. Let DATA be the Element to be deleted. TEMP, HOLD is the temporary pointer to hold the address of the node.

```
1. Input the DATA to be deleted
2. if ((START->DATA)is equal to DATA)
    (a) TEMP = START
    (b) START = STAR->next
    (c) START->prev=NULL
    (d) set free the node TEMP , which is deleted
    (e) Exit
3. HOLD = START
4. while((HOLD->next->next) not equal to NULL)
    (a)if (HOLD->next->DATA) equal to DATA)
        (a1) TEMP = HOLD->next
        (a2)HOLD->next=TEMP->next
        (a3) TEMP->next->prev=HOLD
        (a4) set free the node TEMP, which is deleted
        (a5) Exit
    (b) HOLD=HOLD->next
5. if ((HOLD->next->DATA)==DATA)
    (a) TEMP=HOLD->next
    (b) set free the node TEMP, which is deleted
    (c) HOLD->next=NULL
    (d) exit
6. Display "DATA not found"
```

```
7.Exit
```

**Algorithm for displaying the doubly linked list**

```
1. if (START is equal to NULL)
    1.a. display "The list is empty"
    1.b. exit
2. TEMP=START
3. while TEMP is not equal to NULL
    3.a. display TEMP->info
    3.b. TEMP=TEMP->next
4. exit
```

**Assignment**

1. write an algorithm and a function to display a doubly linked list in reverse order.
2. write an algorithm and a function to count the number of elements in the doubly linked list.
3. write an algorithm and function for searching a number in doubly linked list.