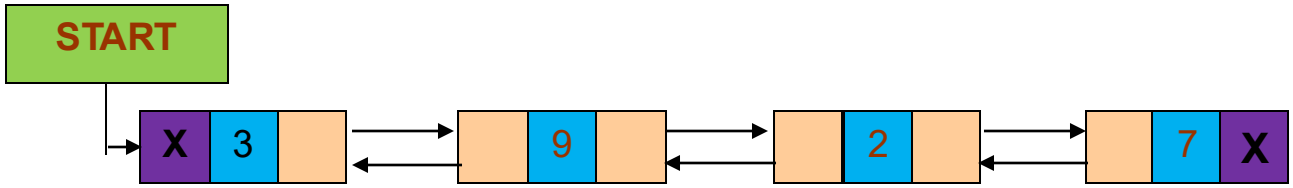
Doubly Linked List

- *A **doubly linked list** or a **two way linked list** is a linked list which contains a pointer to the **NEXT node** as well as **PREVious node** in the sequence.*
- *Therefore, it consists of three parts and not just two.*
- *The three parts are **data, a pointer to the NEXT node and a pointer to the PREVious node***

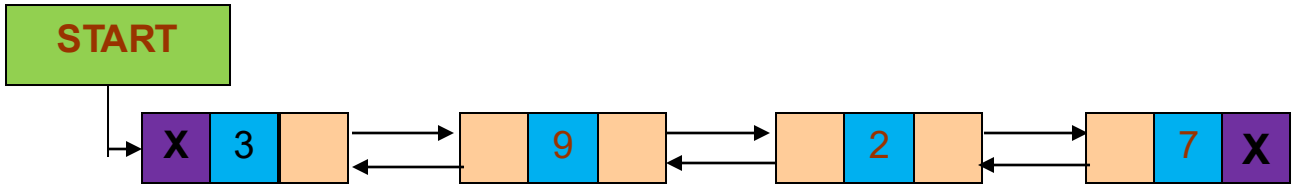


Doubly Linked List

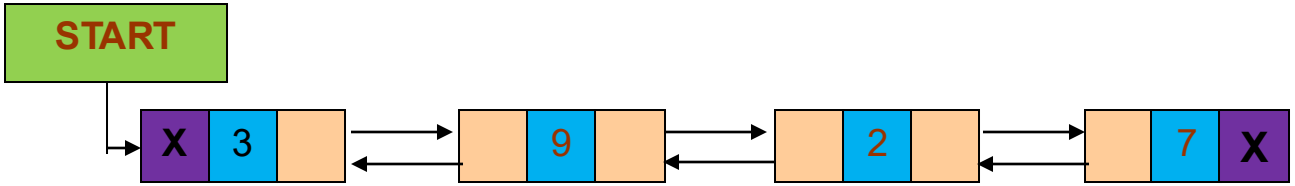
NODE :

```
struct node
{
    struct node *PREV;
    int num;
    struct node *NEXT;
};

struct node *START = NULL;
struct node *NEW_NODE = (struct node*) malloc(sizeof(struct node));
```



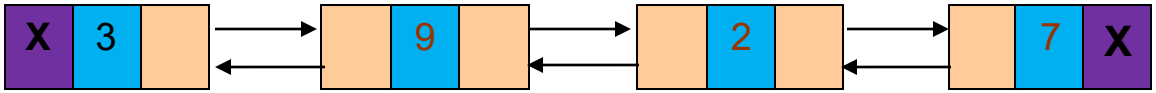
DLL - TRAVERSING A LINKED LIST



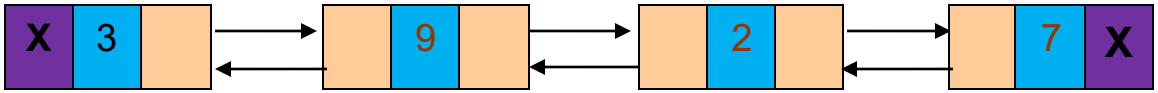
ALGORITHM FOR TRAVERSING A DOUBLY LINKED LIST

```
Step 1: [INITIALIZE] SET PTR = START
Step 2: Repeat Steps 3 and 4 while PTR != NULL
Step 3:              Apply Process to PTR  ->  DATA
Step 4:              SET PTR = PTR  ->  NEXT
        [END OF LOOP]
Step 5: EXIT
```

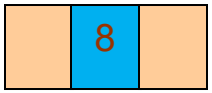
DLL - INSERTING A NODE AT THE BEGINNING



START

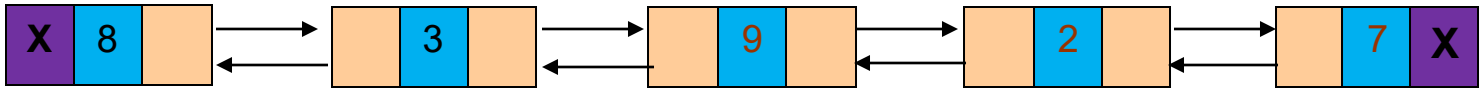


START



NEW NODE

```
SET NEW_NODE -> DATA = VAL
SET NEW_NODE -> NEXT = START
SET NEW_NODE -> PREV = NULL
SET START -> PREV = NEW_NODE
SET START = NEW_NODE
```



START

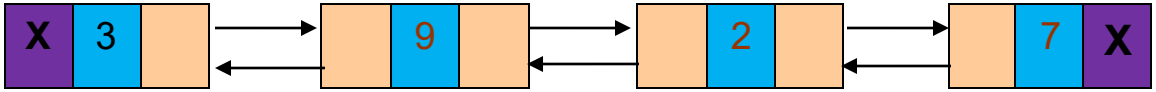
DLL - INSERTING A NODE AT THE BEGINNING

ALGORITHM TO INSERT A NEW NODE IN THE BEGINNING OF THE DOUBLY LINKED LIST

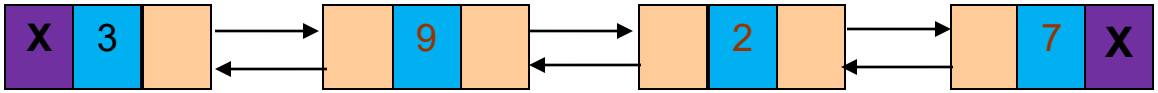
```
Step 1: IF AVAIL = NULL, then
                Write OVERFLOW
                Go to Step 8
        [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET NEW_NODE -> DATA = VAL
Step 4: SET NEW_NODE -> NEXT = START
Step 5: SET NEW_NODE -> PREV = NULL
Step 6: SET START -> PREV = NEW_NODE
Step 7: SET START = NEW_NODE
Step 8: EXIT
```

DLL - INSERTING A NODE AT THE END

```
SET PTR = START
```



START, PTR



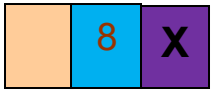
START

PTR

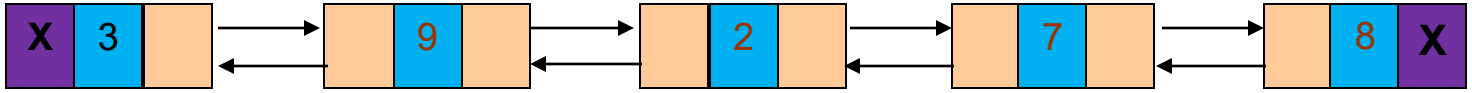
```
SET NEW_NODE -> DATA = VAL

SET PTR -> NEXT = NEW_NODE

SET NEW_NODE -> PREV = PTR
```



NEW NODE



START

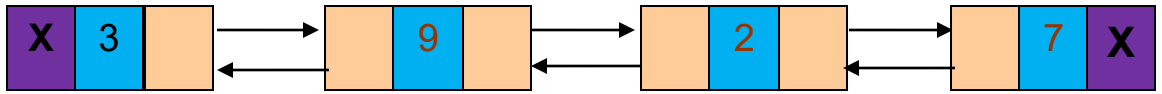
DLL - INSERTING A NODE AT THE END

ALGORITHM TO INSERT A NEW NODE AT THE END OF THE DOUBLY LINKED LIST

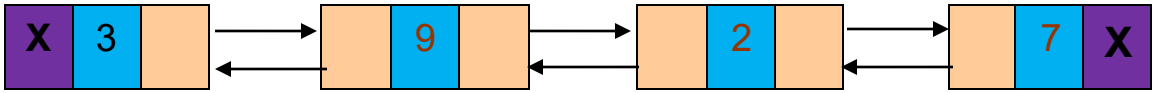
```
Step 1: IF AVAIL = NULL, then
                Write OVERFLOW
                Go to Step 10
         [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET NEW_NODE -> DATA = VAL
Step 4: SET NEW_NODE -> NEXT = NULL
Step 5: SET PTR = START
Step 6: Repeat Step 7 while PTR -> NEXT != NULL
Step 7:           SET PTR = PTR -> NEXT
        [END OF LOOP]
Step 8: SET PTR -> NEXT = NEW_NODE
Step 9: SET NEW_NODE -> PREV = PTR
Step 10: EXIT
```

DLL - INSERTING A NODE AFTER NODE THAT HAS VALUE NUM

```
SET PTR = START
```



START, PTR



START

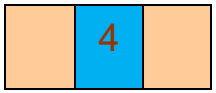
PTR

```
SET NEW_NODE  -> NEXT = PTR  -> NEXT
SET PTR -> NEXT -> PREV  = NEW_NODE
```

PTR

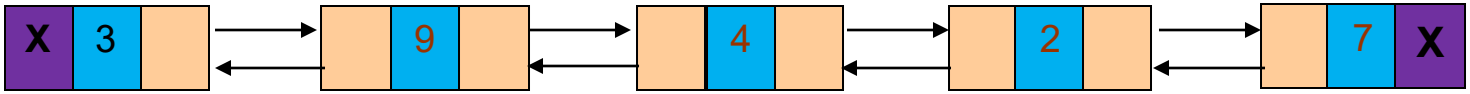
PTR->NEXT

START




```
SET PTR -> NEXT = NEW_NODE
SET NEW_NODE -> PREV = PTR
```

NEW NODE



START

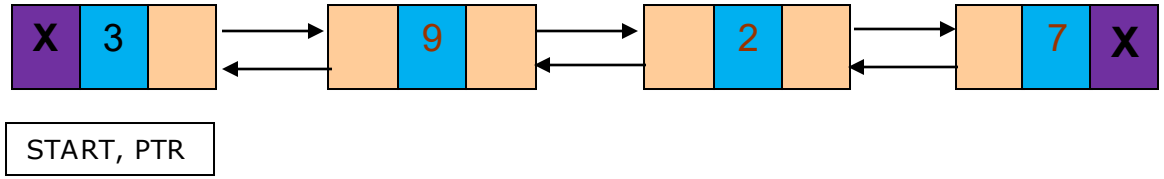
DLL - INSERTING A NODE AFTER A GIVEN NODE

ALGORITHM TO INSERT THE NODE AFTER A GIVEN NODE IN THE DOUBLY LINKED LIST

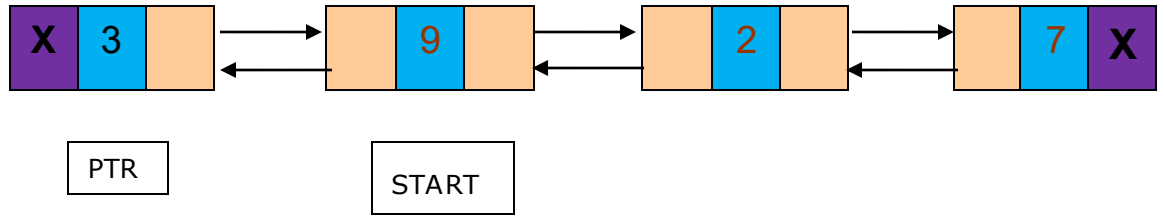
```
Step 1: IF AVAIL = NULL, then
                        Write OVERFLOW
                        Go to Step 11
        [END OF IF]
Step 2: SET NEW_NODE = AVAIL
Step 3: SET NEW_NODE - > DATA = VAL
Step 4: SET PTR = START
Step 5: Repeat Steps 6 while PTR -> DATA  != NUM
Step 6:                 SET PTR = PTR -> NEXT
        [END OF LOOP]
Step 7: SET NEW_NODE  -> NEXT = PTR  -> NEXT
Step 8: SET PTR -> NEXT -> PREV  = NEW_NODE
Step 9: SET PTR -> NEXT = NEW_NODE
Step 10: SET NEW_NODE -> PREV = PTR
Step 11: EXIT
```

DLL - DELETING THE FIRST NODE

SET PTR = START

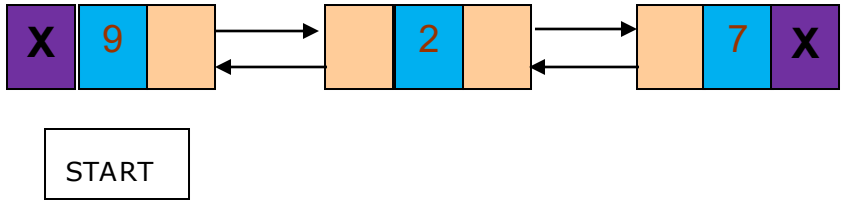


SET START = START -> NEXT



SET START -> PREV = NULL

FREE PTR



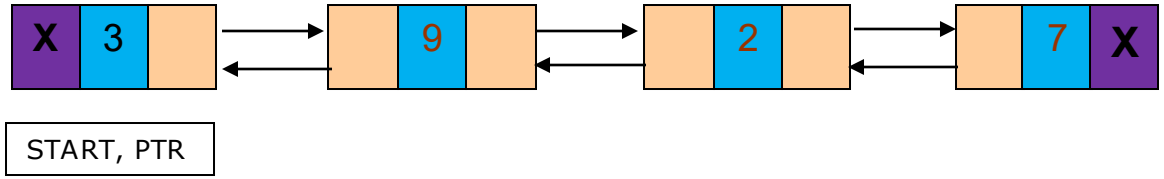
DLL - DELETING THE FIRST NODE

ALGORITHM TO DELETE THE FIRST NODE FROM THE DOUBLY LINKED LIST

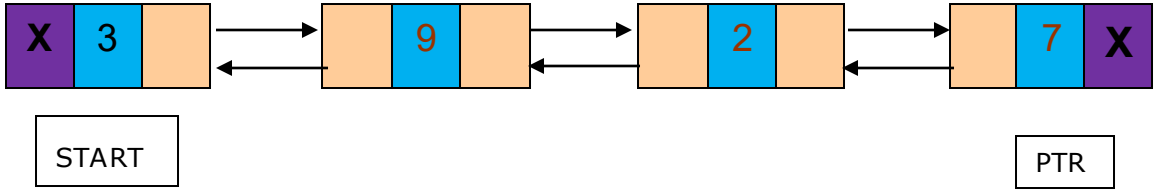
```
Step 1: IF START = NULL, then
                Write UNDERFLOW
                Go to Step 6
        [END OF IF]
Step 2: SET PTR = START
Step 3: SET START = START -> NEXT
Step 4: SET START -> PREV = NULL
Step 5: FREE PTR
Step 6: EXIT
```

DLL - DELETING THE LAST NODE

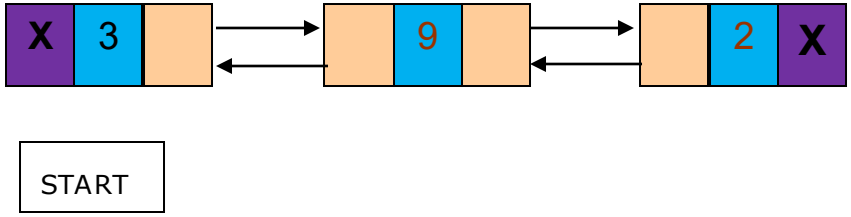
SET PTR = START



SET PTR -> PREV -> NEXT = NULL



FREE PTR



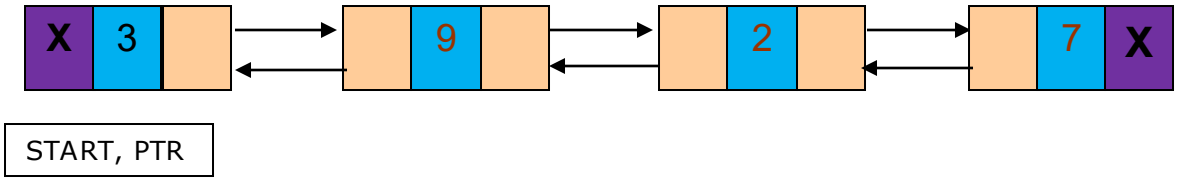
DLL - DELETING THE LAST NODE

ALGORITHM TO DELETE THE LAST NODE OF THE DOUBLY LINKED LIST

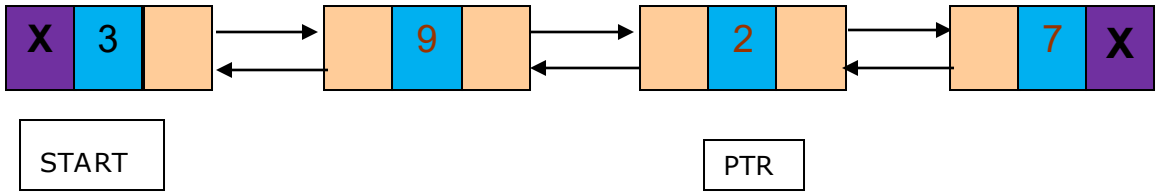
```
Step 1: IF START = NULL, then
                Write UNDERFLOW
                Go to Step 7
          [END OF IF]
Step 2: SET PTR = START
Step 3: Repeat Steps 4 while PTR -> NEXT != NULL
Step 4:                    SET PTR = PTR -> NEXT
         [END OF LOOP]
Step 5: SET PTR -> PREV -> NEXT = NULL
Step 6: FREE PTR
Step 7: EXIT
```

DLL - DELETING THE NODE WHOSE VALUE IS NUM

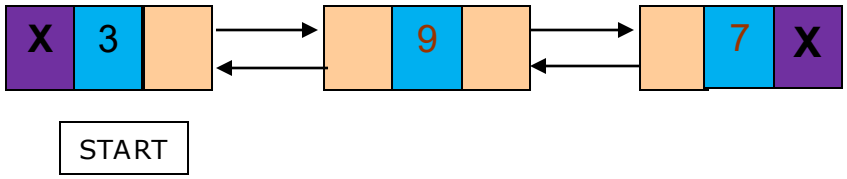
```
SET SET PTR = START
```




```
SET PTR -> PREV -> NEXT = PTR -> NEXT
SET PTR -> NEXT -> PREV = PTR -> PREV
```




```
FREE PTR
```



DLL - DELETING THE NODE WHOSE VALUE IS NUM

ALGORITHM TO DELETE THE NODE WHOSE VALUE IS NUM FROM THE DOUBLY LINKED LIST

```
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 -> DATA != NUM
Step 4:                  SET PTR = PTR -> NEXT
        [END OF LOOP]
Step 5: SET PTR -> PREV -> NEXT = PTR -> NEXT
Step 6: SET PTR -> NEXT -> PREV = PTR -> PREV
Step 7: FREE PTR
Step 8: EXIT
```

SLL - INSERTING A NODE AT THE END

```
IF AVAIL = NULL, then
   Write OVERFLOW
[END OF IF]
SET New_Node = AVAIL
SET New_Node -> DATA = VAL
SET New_Node -> Next = NULL
SET PTR = START
while PTR -> NEXT != NULL
   SET PTR = PTR -> NEXT
[END OF LOOP]
SET PTR -> NEXT = New_Node
EXIT
```

DLL - INSERTING A NODE AT THE END

```
IF AVAIL = NULL, then
   Write OVERFLOW
[END OF IF]
SET New_Node = AVAIL
SET New_Node -> DATA = VAL
SET New_Node -> Next = NULL
SET PTR = START
while PTR -> NEXT != NULL
   SET PTR = PTR -> NEXT
[END OF LOOP]
SET PTR -> NEXT = New_Node
SET NEW_NODE -> PREV = PTR
EXIT
```

SLL - DELETING A NODE AT THE END

```
IF START = NULL, then
   Write UNDERFLOW
[END OF IF]
SET PTR = START
while PTR -> NEXT != NULL
   SET PREPTR = PTR
   SET PTR = PTR -> NEXT
[END OF LOOP]
SET PREPTR -> NEXT = NULL
FREE PTR
EXIT
```

DLL - DELETING A NODE AT THE END

```
IF START = NULL, then
   Write UNDERFLOW
[END OF IF]
SET PTR = START
while PTR -> NEXT != NULL
   SET PTR = PTR -> NEXT
[END OF LOOP]
SET PTR -> PREV -> NEXT =NULL
FREE PTR
EXIT
```