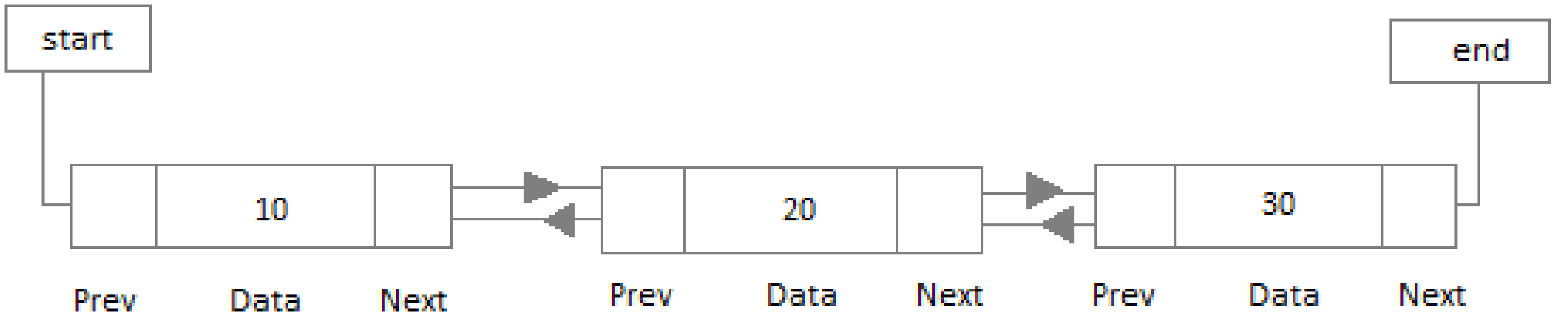
GLS UNIVERSITY

DATA STRUCTURES
UNIT– II

Doubly Linked List

- In a doubly linked list, each node contains a data part and two addresses, one for the previous node and one for the next node.



Doubly Linked List

Advantages:

- A DLL can be traversed in the both forward and backward direction.
- The delete operation in DLL is more efficeint if pointer to the node to be deleted is given.

Disadvantages:

- Every node of DLL require extra space for an previous pointer.
- All operation require an extra pointer previous to be maitained.

Creating a Node

- A node is the main structure of the linked list.
- It contains all details about the data and the address of previous node and next node.
- A node can contain many data fields and many address fields, but should contain at least one address field.
- Previous Address part of first node contains NULL value specifying starting of the list.
- Next Address part of last node contains a NULL value specifying end of the list.
- A node can be represented either by structures (struct in C or C++) or by classes (class in C++ or Java)

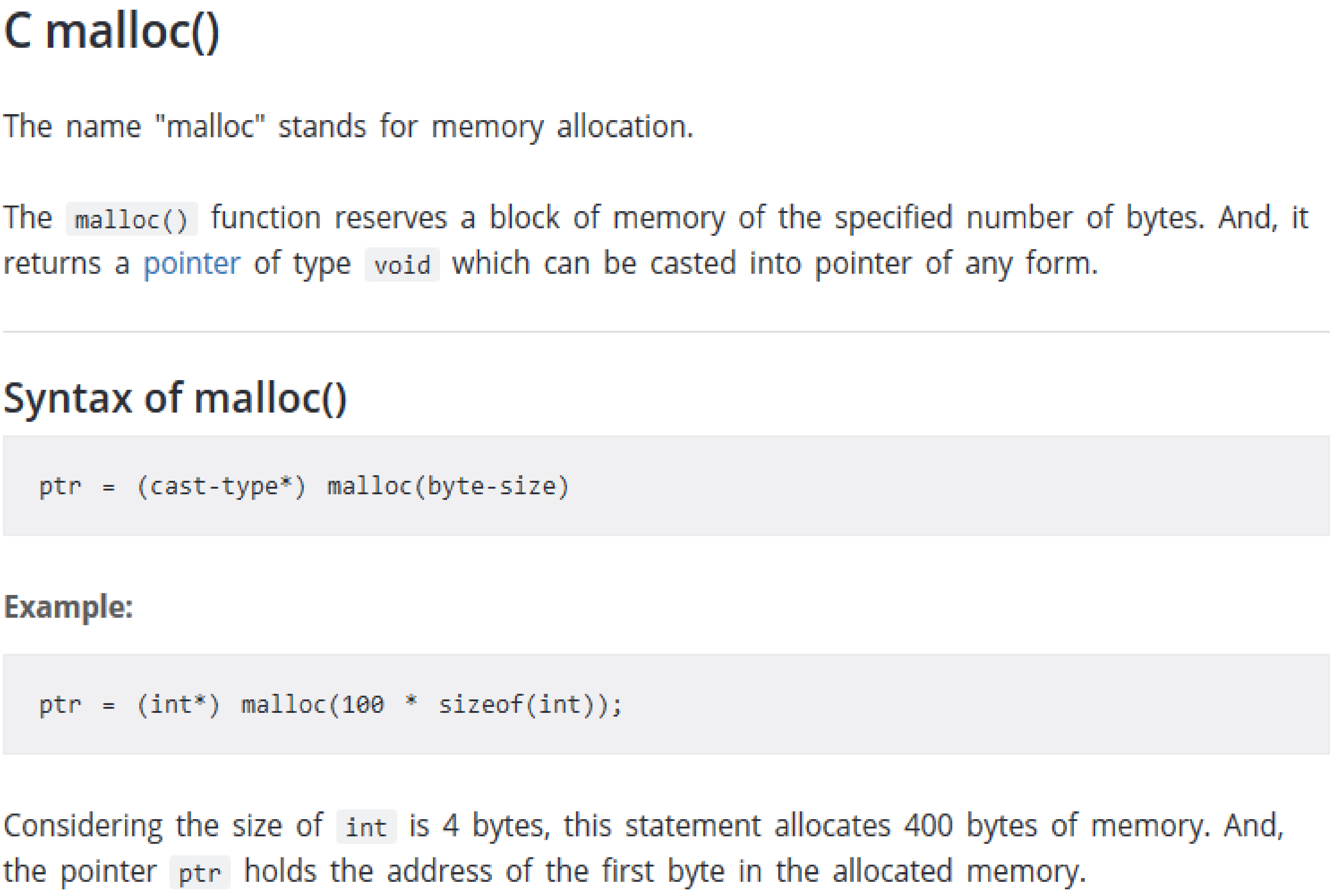
Creating a Node

- A basic structure of a node in C programming language.

- // Basic structure of a node
- struct node
- {

  struct node *prev; // address of previous node
- int data;   // Data
- struct node * next; // Address of next node
- };

Creating a Node



C malloc()

The name "malloc" stands for memory allocation.

The `malloc()` function reserves a block of memory of the specified number of bytes. And, it returns a pointer of type `void` which can be casted into pointer of any form.

Syntax of malloc()

```
ptr = (cast-type*) malloc(byte-size)
```

Example:

```
ptr = (int*) malloc(100 * sizeof(int));
```

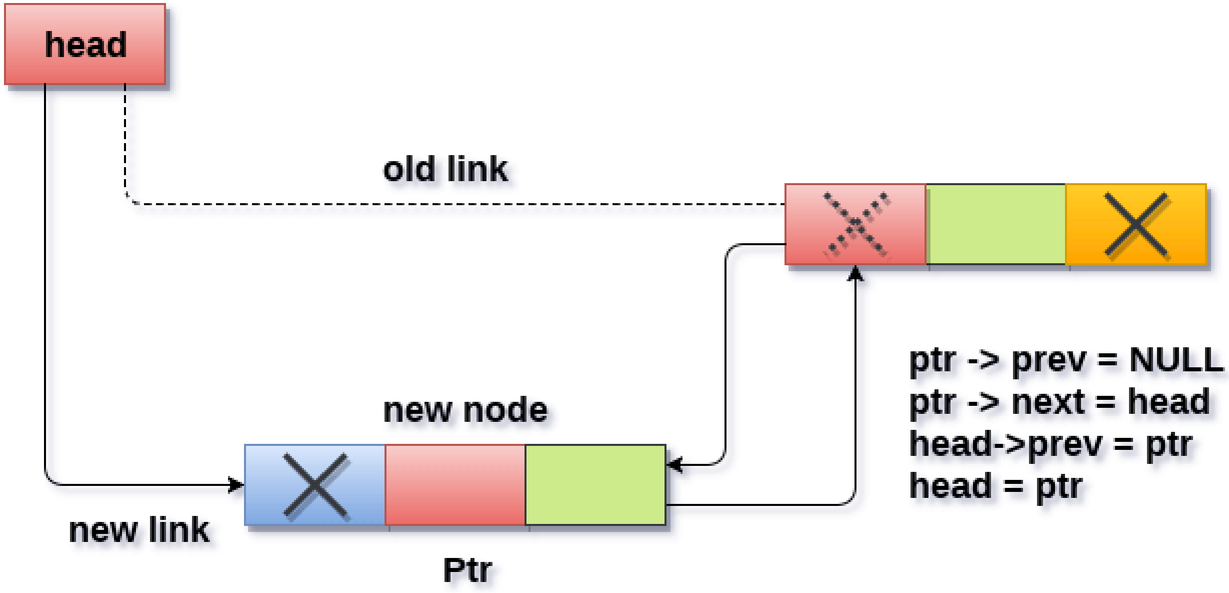
Considering the size of `int` is 4 bytes, this statement allocates 400 bytes of memory. And, the pointer `ptr` holds the address of the first byte in the allocated memory.

Inserting a node in a Doubly linked list

A node can be added in three ways

1) the new node is inserted at the beginning.
2) the new node is inserted at the end.
3) the new node is inserted after a given node.

Inserting a node at the beginning in the double LL



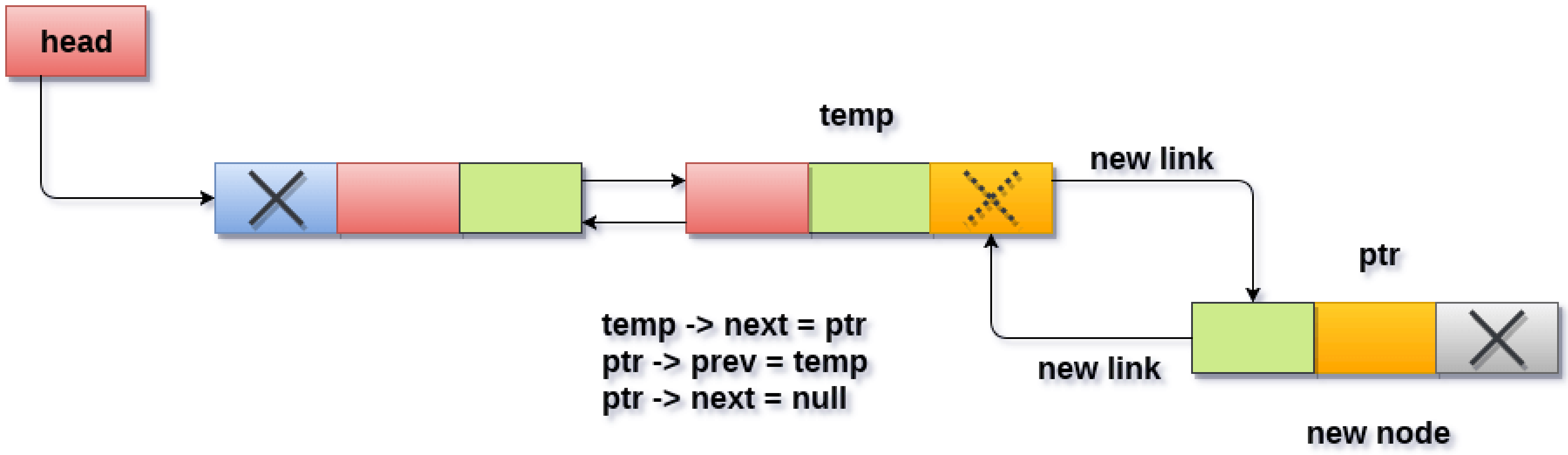
ptr -> prev = NULL
ptr -> next = head
head->prev = ptr
head = ptr

Insertion into doubly linked list at beginning

Algorithm to insert a node in starting

```
Step1: SET New_Node = AVAIL (malloc)
Step2: SET New_Node -> DATA = VAL
Step3: SET New_Node -> PREV= NULL
Step4: SET New_Node -> Next = START
Step 5: SET START-> PREV = New_Node
Step6: SET START = New_Node
Step7: EXIT
```

Inserting a node at the ending in the double LL



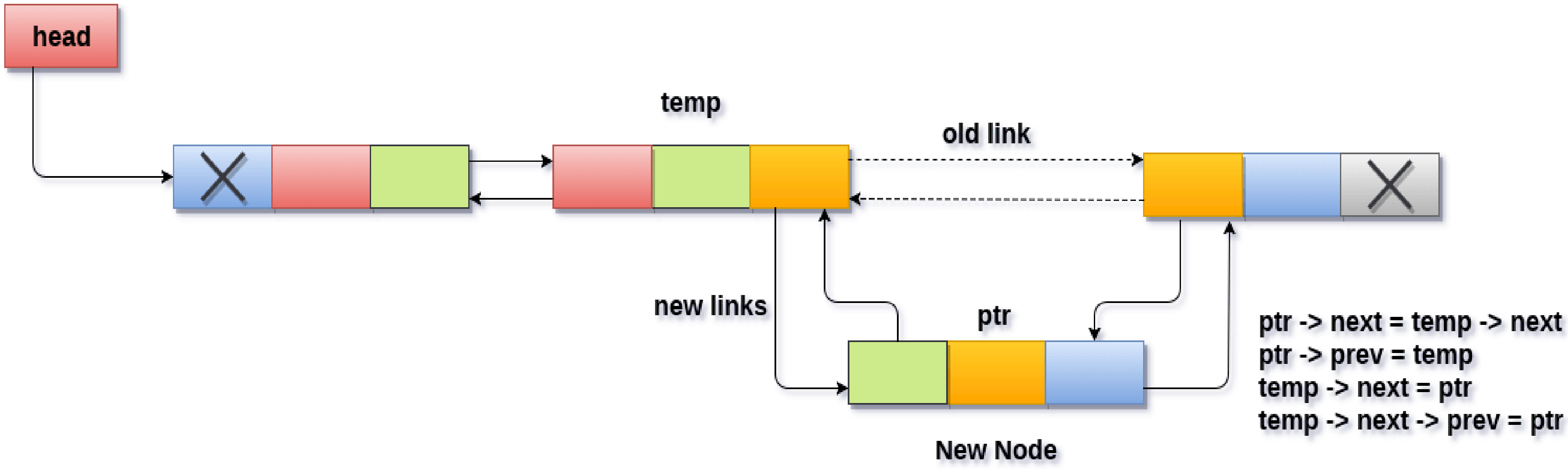
temp -> next = ptr
ptr -> prev = temp
ptr -> next = null

Insertion into doubly linked list at the end

Algorithm to insert a node in the end

```
Step1: SET New_Node = AVAIL
Step2: SET New_Node -> DATA = VAL
Step3: SET New_Node -> Next = NULL
Step4: SET PTR = START
Step5: Repeat Step 6 while PTR -> NEXT != NULL
Step6:     SET PTR = PTR -> NEXT
    [END OF LOOP]
Step7: SET PTR->NEXT = NEW_NODE
Step8: New_Node -> PREV = PTR
Step9: EXIT
```

Inserting a node after a given node in the doubly LL



```
ptr -> next = temp -> next
ptr -> prev = temp
temp -> next = ptr
temp -> next -> prev = ptr
```

Insertion into doubly linked list after specified node

Algorithm to insert a node after given node

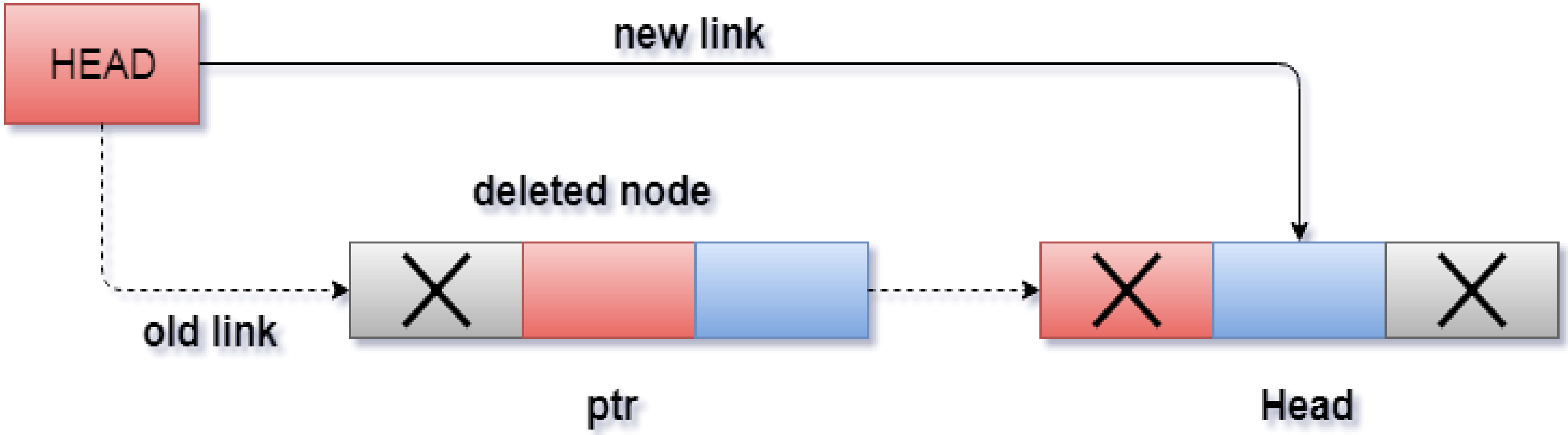
```
Step1: SET New_Node = AVAIL
Step2: SET New_Node -> DATA = VAL
Step3: SET PTR = START
Step4: Repeat Step 5 while PTR -> DATA != NUM
Step5:     SET PTR = PTR -> NEXT
Step6: SET New_Node -> NEXT = PTR -> NEXT
Step7: SET New_Node -> PREV = PTR
Step8: SET PTR->NEXT->PREV = New_Node
Step9: SET PTR-> NEXT =  New_Node
Step9: EXIT
```

Deleting a node from Doubly linked list

A node can be deleted in three ways

1) The first node is deleted
2) The last node is deleted
3) The node after a given node is deleted.

Delete the first node in the Doubly linked list



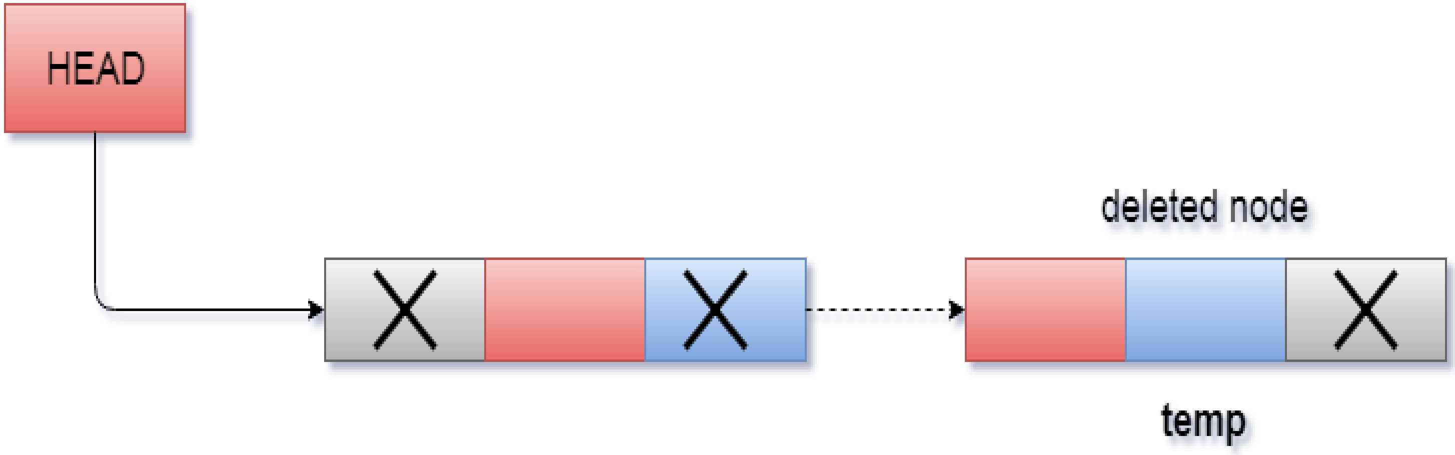
```
ptr = head
head = head -> next
head -> prev = NULL
free (ptr)
```

Deletion in doubly linked list from beginning

Delete the first node in the Doubly linked list

```
Step1: IF START = NULL then
    Write UNDERFLOW
    Goto Step6
    [END OF IF]
Step2: SET PTR = START
Step3:  SET START= START->NEXT
Step4:  SET START -> PREV = NULL
Step5:  FREE PTR
Step6:  EXIT
```

Delete the last node in the Doubly linked list



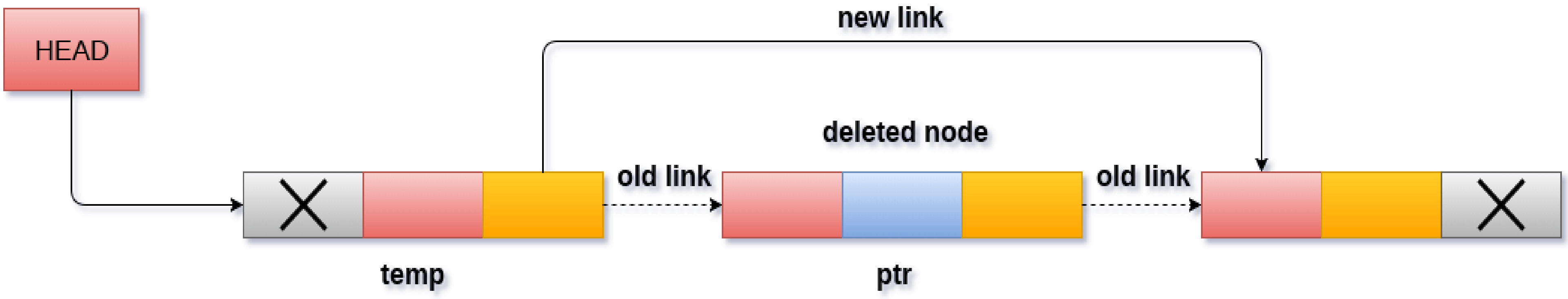
```
temp->prev->next = NULL
free(temp)
```

Deletion in doubly linked list at the end

Delete the last node in the Doubly linked list

```
Step1: IF START = NULL then
    Write UNDERFLOW
    Goto Step7
    [END OF IF]
Step2: SET PTR = START
Step3:  Repeat Step 4 while PTR->NEXT !=NULL
Step4:          SET PTR=PTR->NEXT
        [END OF LOOP]
Step5:  SET PTR -> PREV -> NEXT = NULL
Step6:  FREE PTR
Step7:  EXIT
```

Delete the middle node in the linked list



temp -> next = ptr -> next
ptr ->next -> prev = temp
free(ptr)

Deletion of a specified node in doubly linked list

Delete the middle node in the linked list

```
Step1: IF START = NULL then
    Write UNDERFLOW
    Goto Step8
    [END OF IF]
Step2: SET PTR = START
Step3:  Repeat Step 4 while PTR->DATA!=NUM
Step4:      SET PTR=PTR->NEXT
        [END OF LOOP]
Step5:  SET TEMP = PTR -> PREV
Step6:  SET TEMP -> NEXT = PTR -> NEXT
```

Delete the middle node in the linked list

```
Step7:  SET PTR -> NEXT -> PREV = TEMP
Step8: FREE PTR
Step9: EXIT
```