Linked List

Outline

- Linked List
- Singly Linked List
- Doubly Linked list
- Circular linked list
- Linked implementation of Stack
- Linked implementation of Queue
- Applications of linked list

Linked List

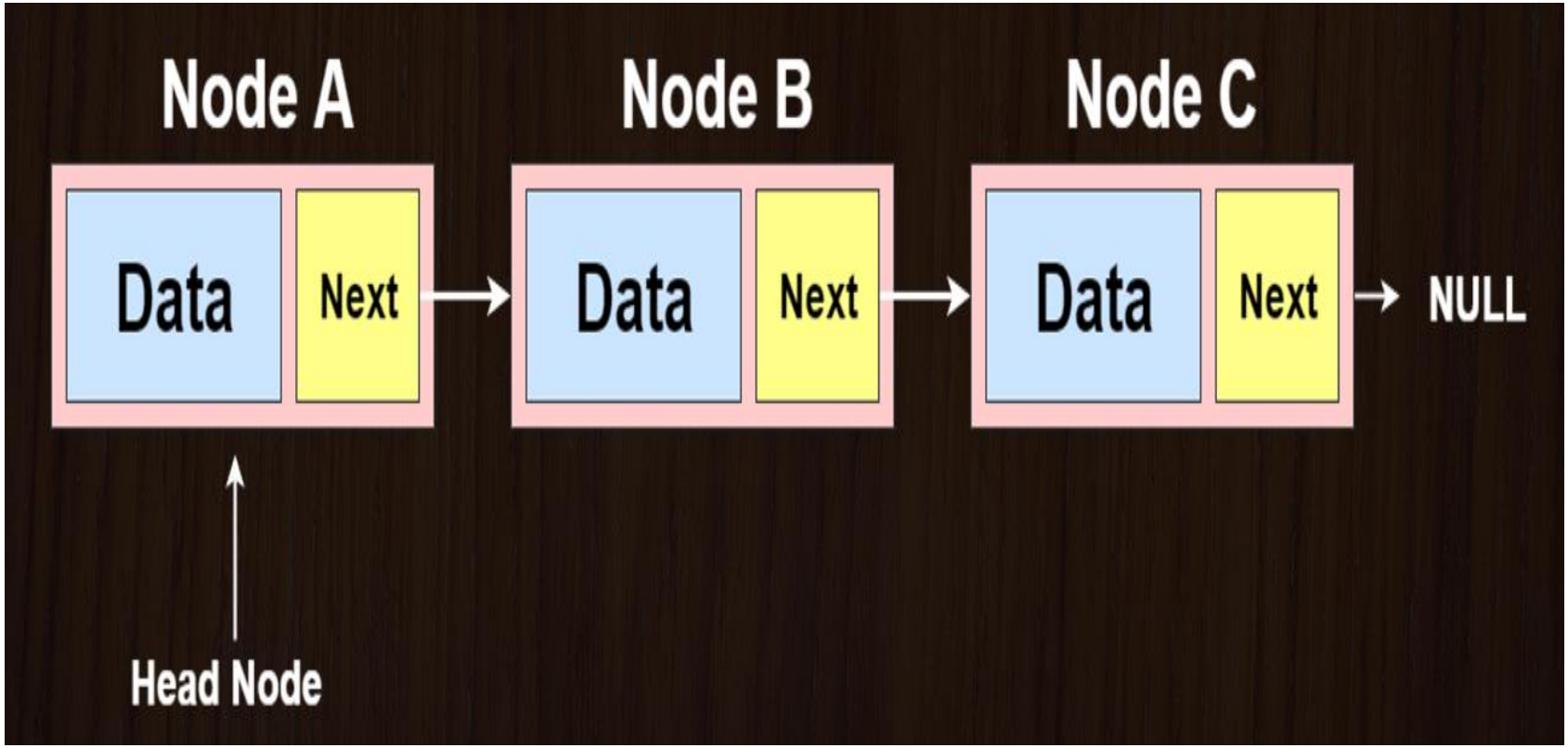
- A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations.
- A Linked List is an ordered collection of data in which each element contains the location(address) of the next element: that is, each element contains two parts : data and link.

Linked List

- In simple words, a linked list consists of **Nodes**. where each node contains:

1. **Data Field :** data stored at that particular address.
2. **Link:** pointer which contains the address of the next node in the memory.

Linked List


Node A
Node B
Node C
Data
Next
Data
Next
Data
Next
NULL
Head Node

Operations on Linked List

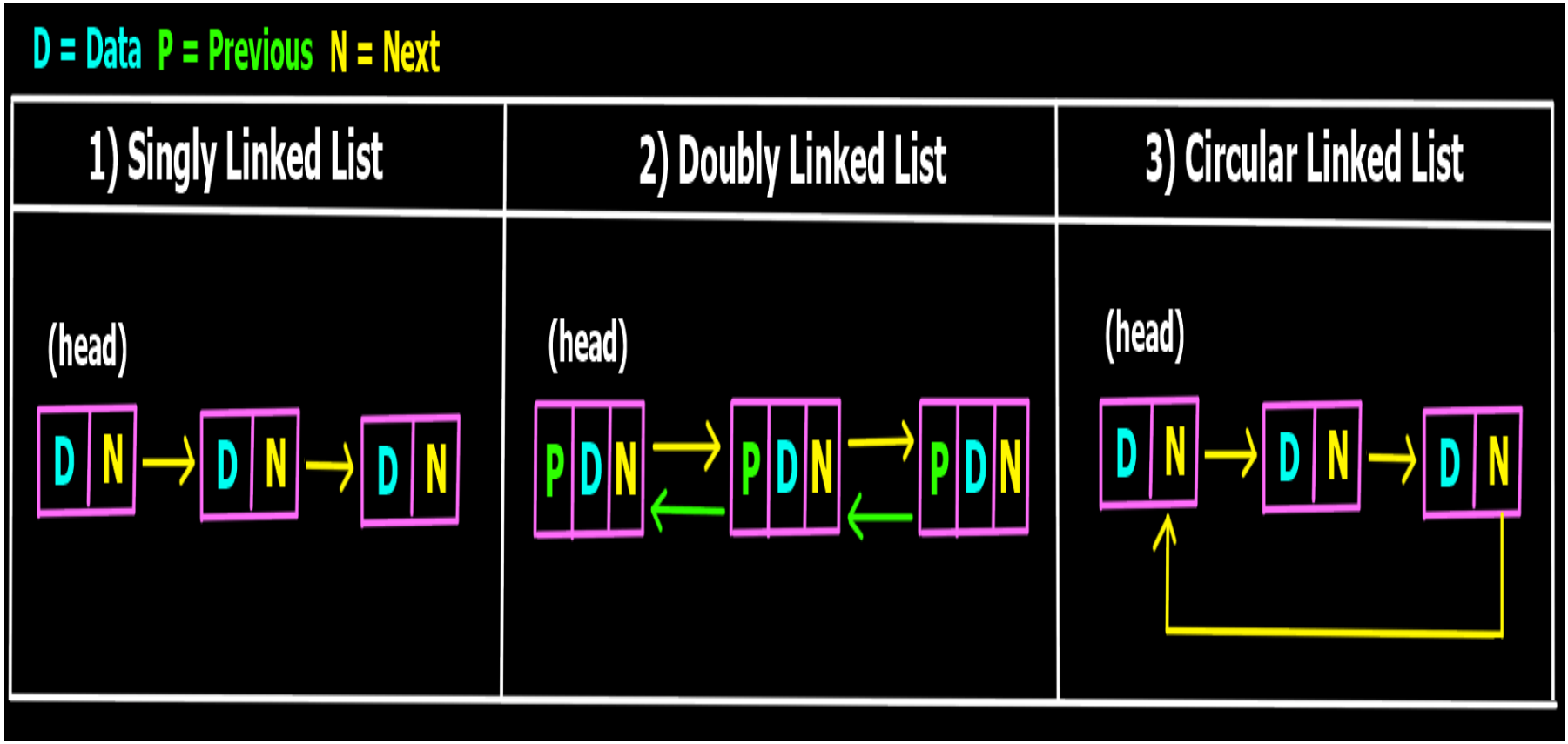
1. **Traverse:** Iterate through the nodes in the linked list starting from the head node.
2. **Append:** Attach a new node (to the end) of a list.
3. **Prepend:** Attach a new node (to the beginning) of the list.
4. **Insert:** attach a new node to a specific position on the list.
5. **Delete:** Remove/Delink a node from the list.
6. **Count:** Returns the no of nodes in linked list.

Types of Linked List

- There are three types of Linked List:

1. Singly Linked List
2. Doubly Linked List
3. Circular Linked List

Types of Linked List



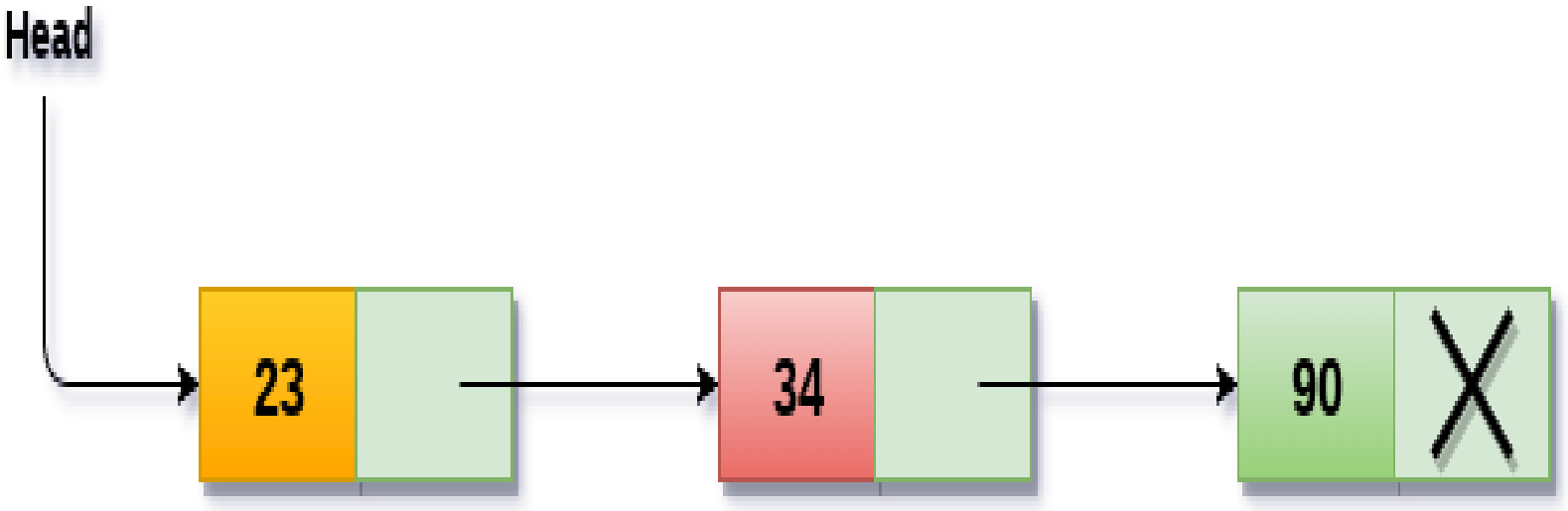
Advantage of link list

- **Advantage:** Insertion and deletion of elements can be efficiently done.
- No wastage of memory.
- **Disadvantage:** Does not support random or direct access.

Singly linked list

- Singly linked list can be defined as the collection of ordered set of elements.
- The number of elements may vary according to need of the program.
- A node in the singly linked list consist of two parts: data part and link part.
- Data part of the node stores actual information that is to be represented by the node while the link part of the node stores the address of its immediate successor.

Singly linked list



Operations on Singly Linked List

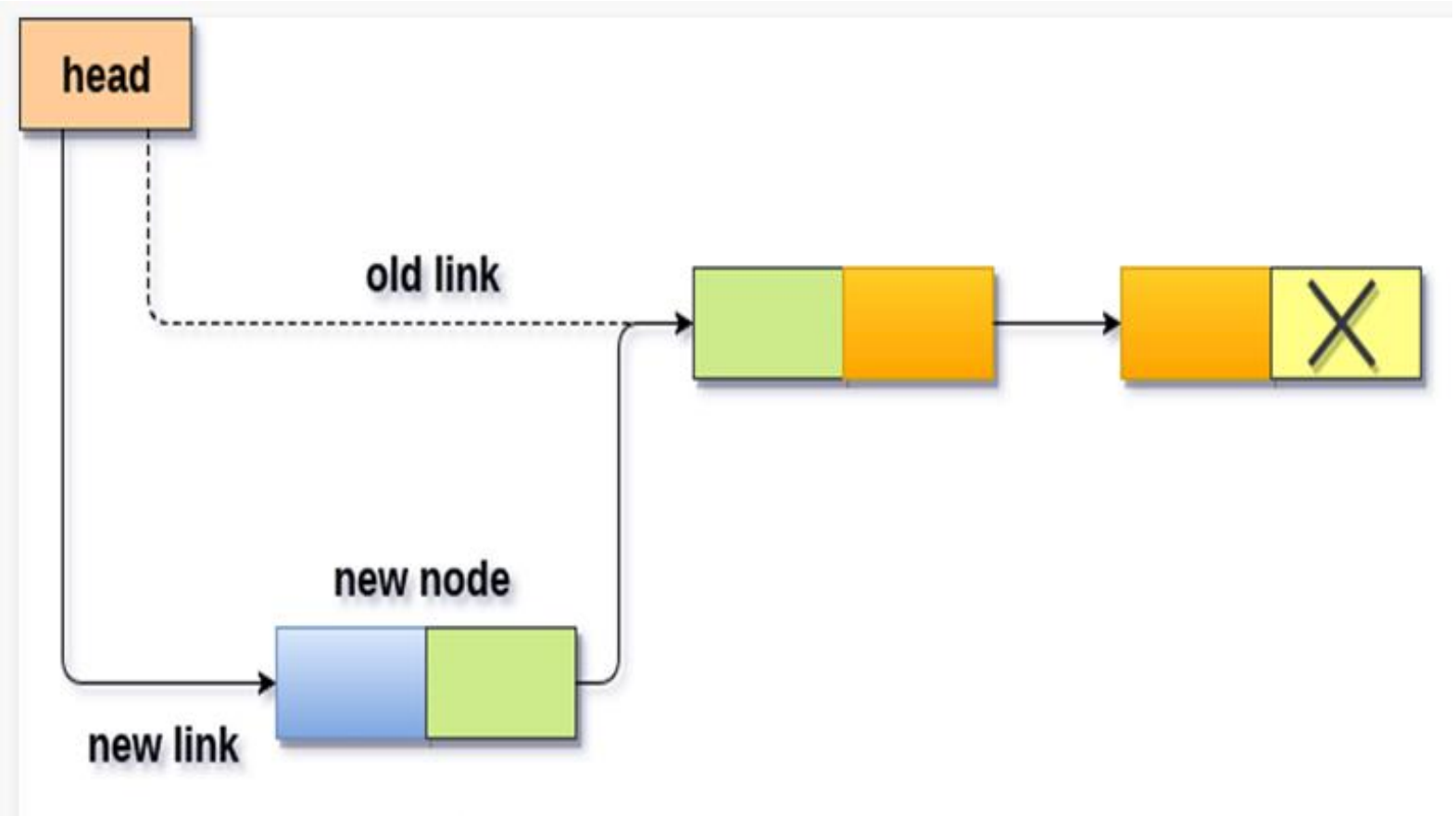
```
struct node
{
    int data;
    struct node *next;
};
struct node *head, *temp;
```

Insertion in singly linked list at beginning

Step 1: IF NEW_NODE = NULL
Write OVERFLOW
Go to Step 5
[END OF IF]

Step 2: SET NEW_NODE → DATA = VAL
Step 3: SET NEW_NODE → NEXT = HEAD
Step 4: SET HEAD = NEW_NODE
Step 5: EXIT

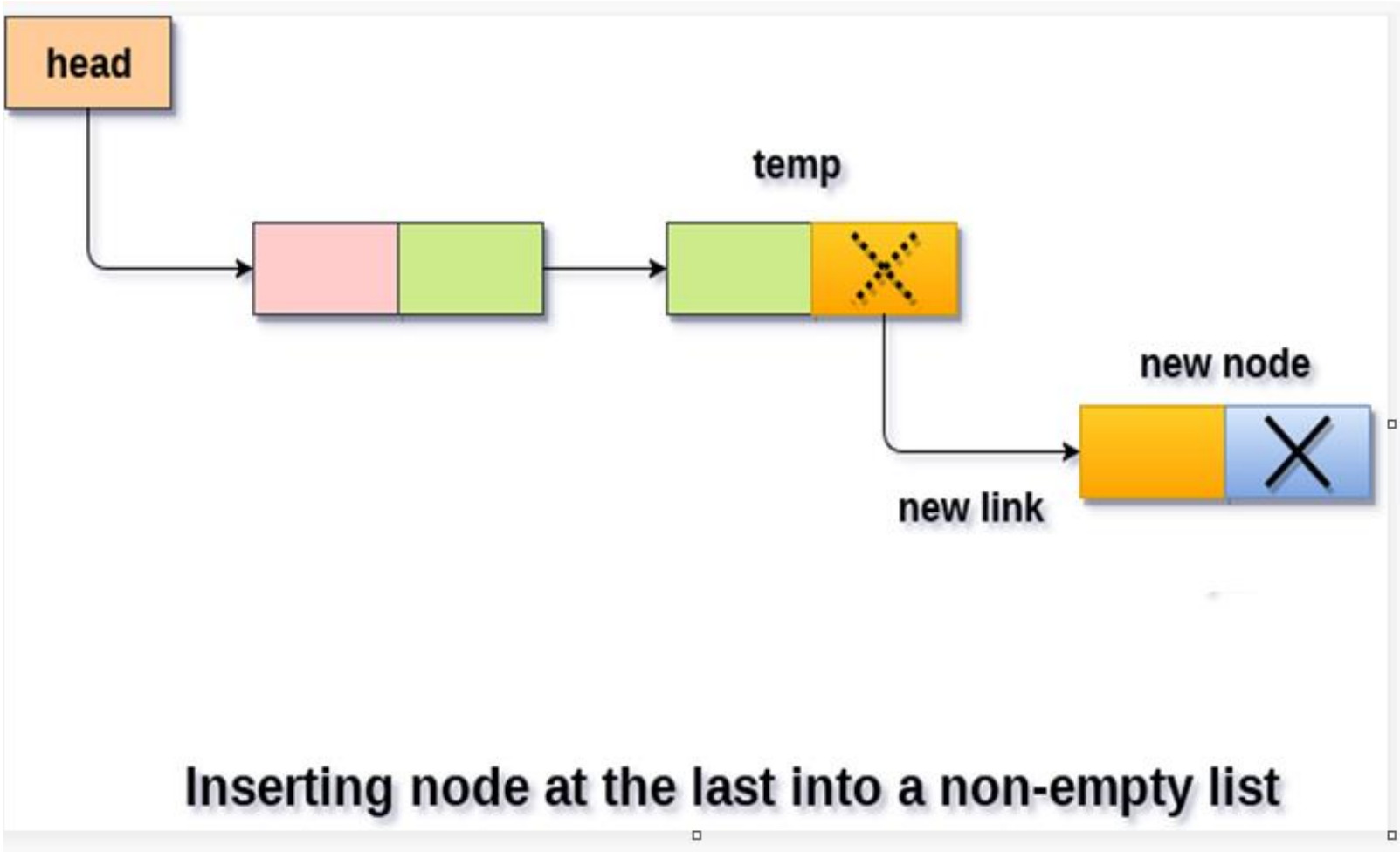
Insertion in singly linked list at beginning



Insertion in singly linked list at the end

Step 1: IF NEW_NODE = NULL Write OVERFLOW
    Go to Step 8
   [END OF IF]
Step 2: SET NEW_NODE - > DATA = VAL
Step 3: SET NEW_NODE - > NEXT = NULL
Step 4: SET PTR = HEAD
Step 5: Repeat Step 6 while PTR - > NEXT != NULL
Step 6: SET PTR = PTR - > NEXT
          [END OF LOOP]
Step 7: SET PTR - > NEXT = NEW_NODE
Step 8: EXIT

Insertion in singly linked list at the end

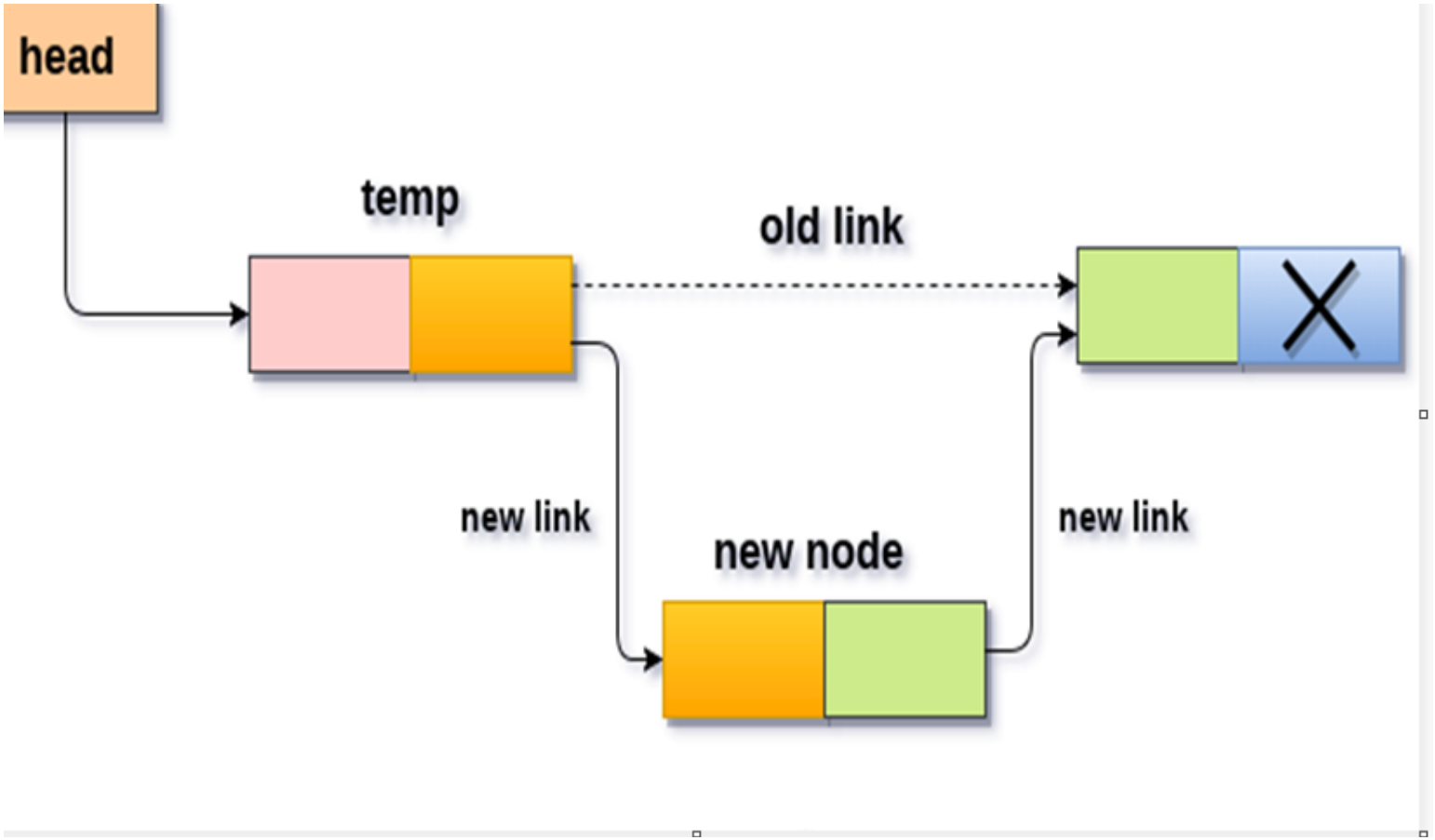


Insertion in singly linked list after specified Node(Location)

```
STEP 1: IF NEW_NODE  = NULL
WRITE OVERFLOW
    GOTO STEP 10
   END OF IF
STEP 2: NEW_NODE → DATA = VAL
STEP 3: SET TEMP = HEAD
STEP 4: SET I = 0
STEP 5: REPEAT STEP 6 AND 7 UNTIL I<POSITION
STEP 6: TEMP = TEMP → NEXT
STEP 7: IF TEMP = NULL
WRITE "DESIRED NODE NOT PRESENT"
     GOTO STEP 10
    END OF IF
                I=I+1;
 END OF LOOP

STEP 8: NEW_NODE → NEXT = TEMP → NEXT
STEP 9: TEMP → NEXT = NEW_NODE
STEP 10: EXIT
```

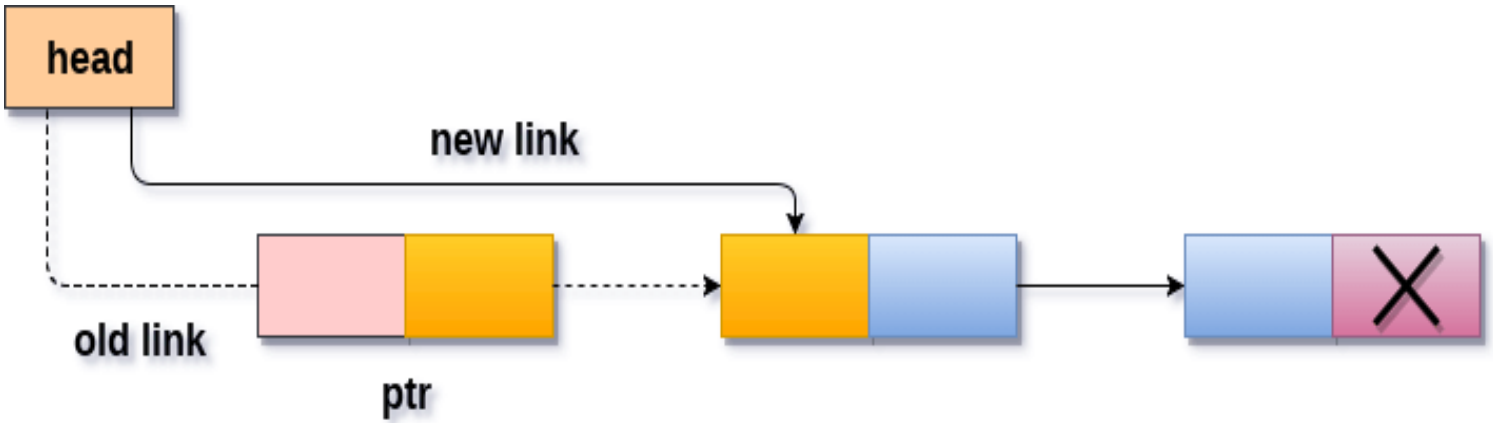
Insertion in singly linked list after specified Node



Deletion in singly linked list at beginning

Step 1: IF HEAD = NULL
Write UNDERFLOW
Go to Step 5
[END OF IF]
Step 2: SET PTR = HEAD
Step 3: SET HEAD = HEAD -> NEXT
Step 4: FREE PTR
Step 5: EXIT

Deletion in singly linked list at beginning



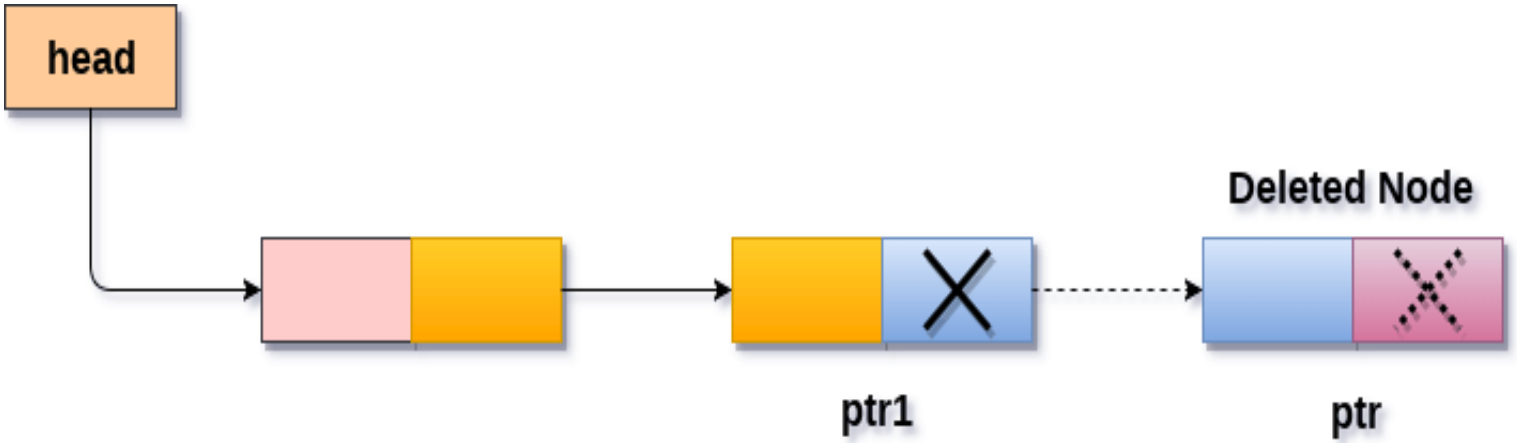

```
ptr = head
head = ptr -> next
free(ptr)
```

Deleting a node from the beginning

Deletion in singly linked list at the end

Step 1: IF HEAD = NULL
Write UNDERFLOW
Go to Step 8
[END OF IF]
Step 2: SET PTR = HEAD
Step 3: Repeat Steps 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

Deletion in singly linked list at the end

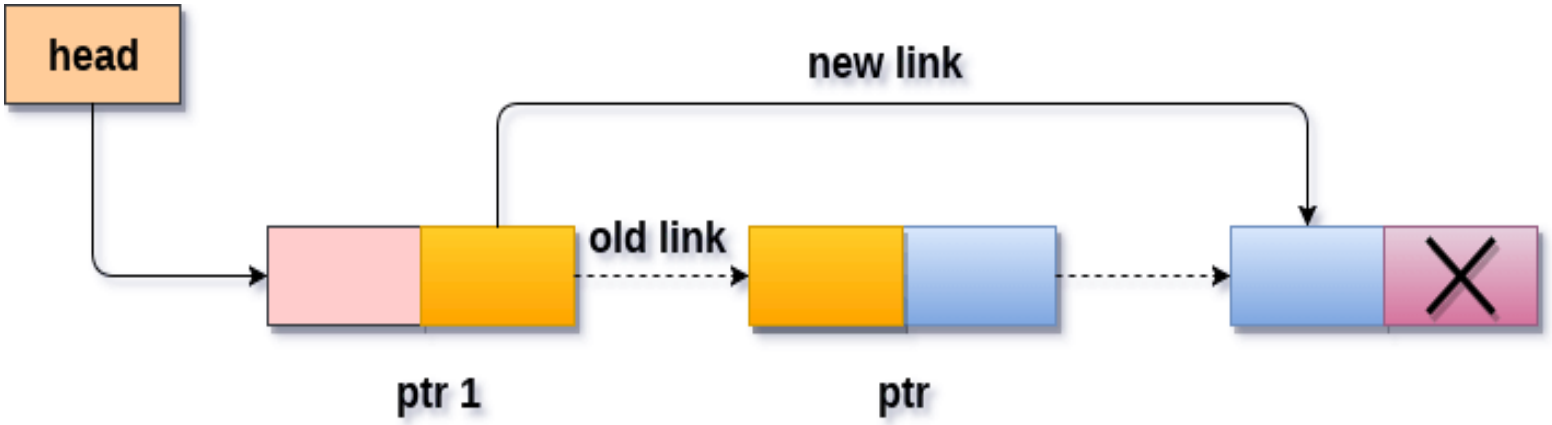


Deleting a node from the last

Deletion in singly linked list after the specified node

```
STEP 1: IF HEAD = NULL
            WRITE UNDERFLOW
            GOTO STEP 10
            END OF IF
STEP 2: SET TEMP = HEAD
STEP 3: SET I = 0
STEP 4: REPEAT STEP 5 TO 8 UNTIL I
STEP 5: TEMP1 = TEMP
STEP 6: TEMP = TEMP → NEXT
STEP 7: IF TEMP = NULL
            WRITE "DESIRED NODE NOT PRESENT"
            GOTO STEP 12
            END OF IF
STEP 8: I = I+1
END OF LOOP
STEP 9: TEMP1 → NEXT = TEMP → NEXT
STEP 10: FREE TEMP
STEP 11: EXIT
```

Deletion in singly linked list after the specified node




```
ptr1 -> next = ptr -> next
free(ptr)
```

Deletion a node from specified position

THANK YOU