Data Structures and Algorithm Analysis

4

Dr. Syed Asim Jalal
Department of Computer Science
University of Peshawar

```c
#include <stdio.h>
#include <stdlib.h>

struct llist {
 char *name;
 struct llist *next;
};

void main (void)
{

struct llist *listItem;
struct llist *head = NULL;
struct llist *temp;



listItem = malloc(sizeof(struct llist));
listItem->name = "Asim";
listItem->next=NULL;

head = listItem;
```

```c
temp = listItem;
listItem = malloc(sizeof(struct llist));
listItem->name = "Computer Science";
listItem->next=NULL;
temp->next = listItem;

temp = listItem;
listItem = malloc(sizeof(struct llist));
temp->next = listItem;
listItem->name = "University of Peshawar";
listItem->next=NULL;

//printing linked list
temp = head;
    while( temp->next != NULL)
    {
        printf("%s \n", temp->name);
        temp = temp->next;
    }
    printf("%s \n", temp->name);
}
```

Traversing a Linked List

Suppose **START** points to the first node in the linked list. The following algorithm will visit all nodes from the START node to the end.

1. If (START is equal to NULL)
   (a) Display “The list is Empty”
   (b) Exit
2. Initialize TEMP = START
3. Repeat step 4 and 5 until (TEMP is equal to NULL)
   4. Display “TEMP →DATA”
   5. TEMP ← TEMP →Next
6. Exit

Algorithm for searching a value

Suppose **START** is the address of the first node in the linked list and **DATA** is the information to be searched. After searching, if the DATA is found. **Found** pointer will contain address of the found node.

1. Input the VALUE to be searched
2. Initialize TEMP = START

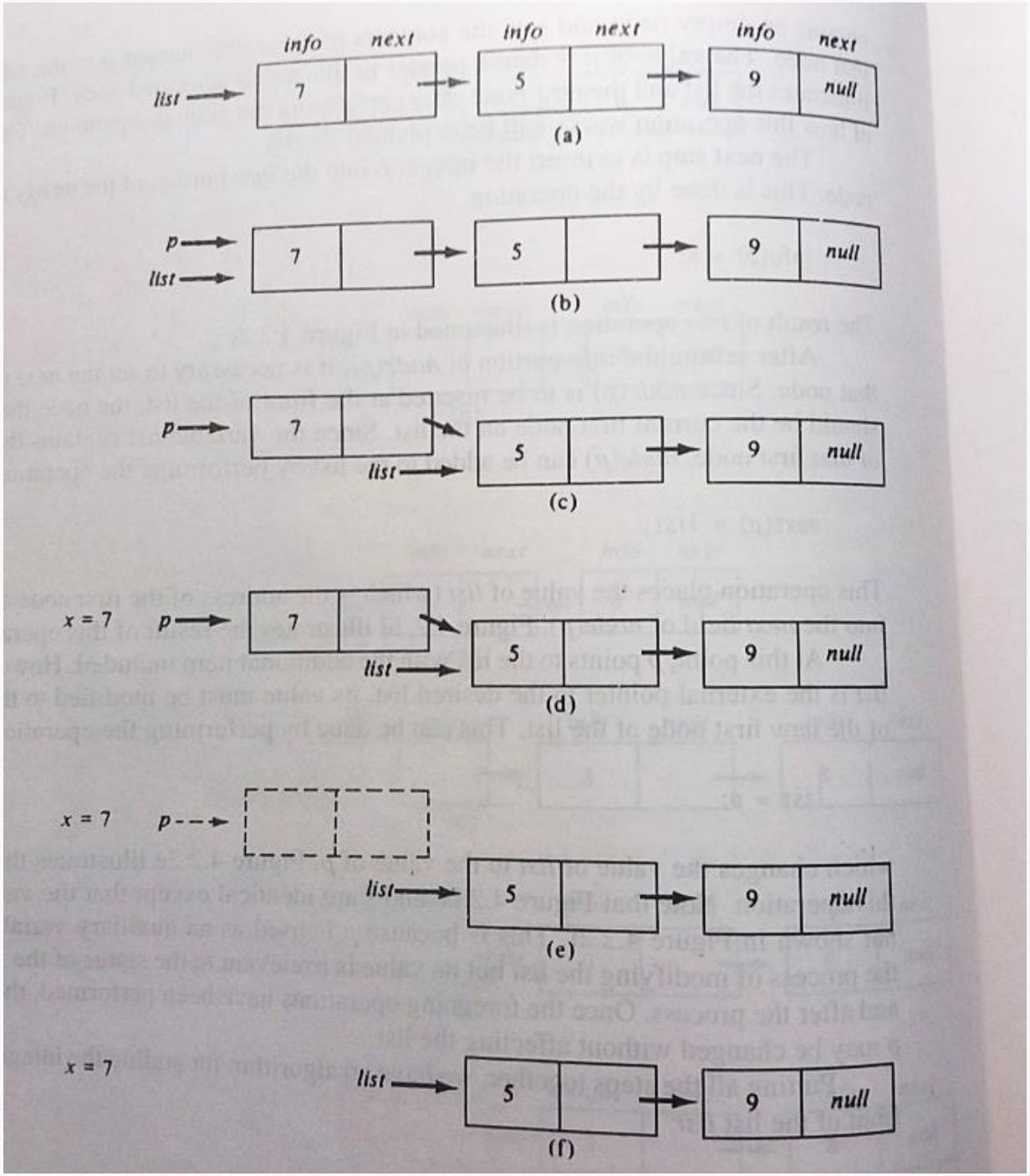
3. Repeat step 4, 5 until (TEMP is equal to NULL)
    4. If (TEMP →DATA is equal to VALUE)
        Found = TEMP
        Exit
    5. TEMP = TEMP →Next

6. If (TEMP is equal to NULL)
    Display “The data is not found in the list”
8. Exit

Algorithm for deleting a node

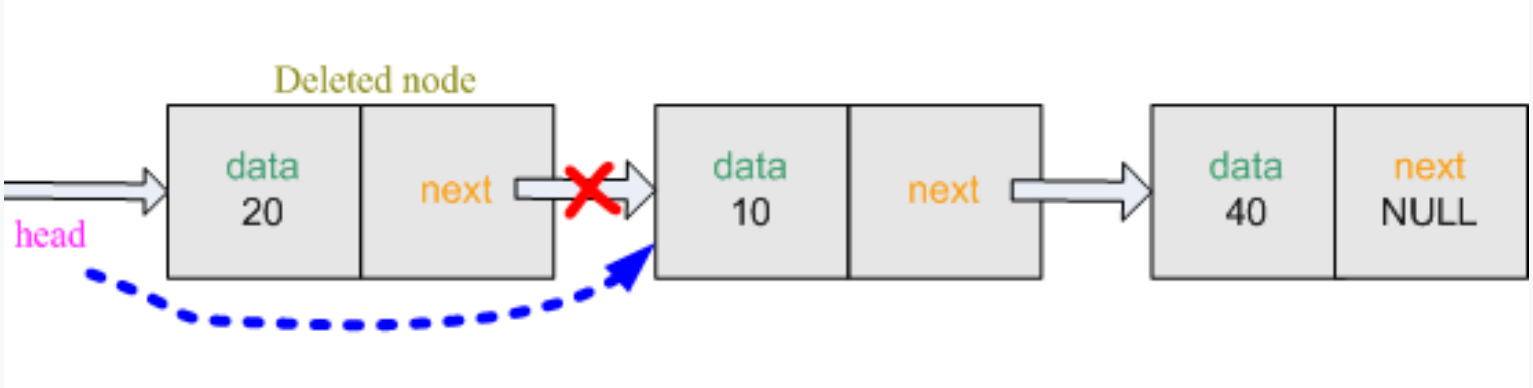
- We can Delete nodes in the Linked List based on the value or information contained in a node.

- There are three possibilities:
  - The node to be deleted is the first node in the linked list
  - The node to be deleted is the last node in the linked list
  - The node to be deleted is in the middle of the linked list

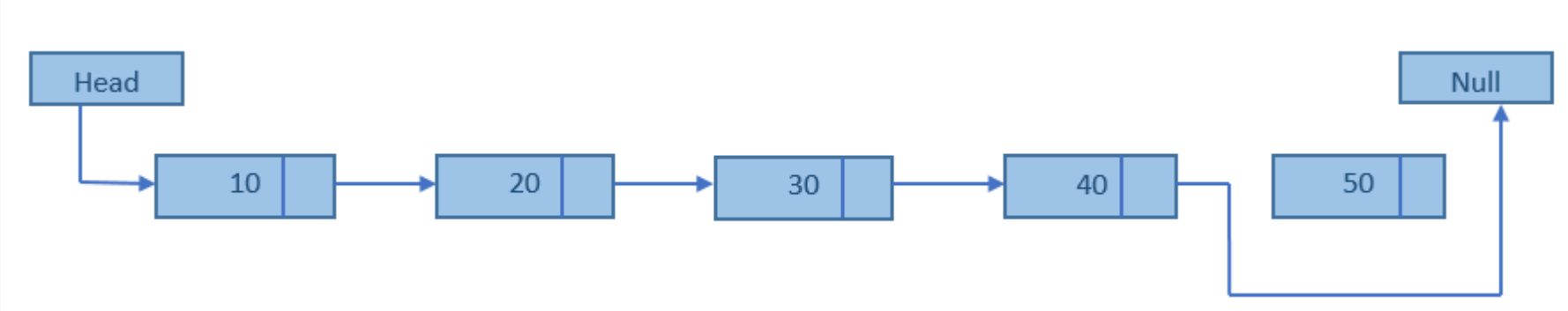


Steps involved in Delete node from **front of a linked List**.

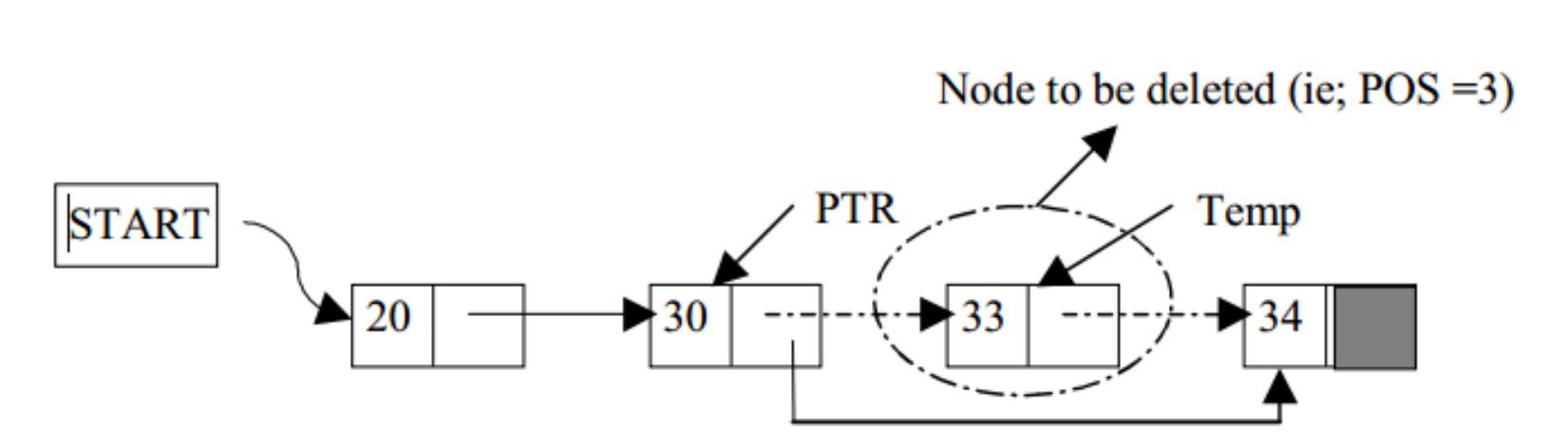
- Node to be deleted is in the start



- Node to be deleted is the Last node



- Node to be deleted is in the middle of the linked list



- Suppose START is the first position in linked list. Let DATA be the element to be deleted. TEMP and PTR are temporary pointers to hold the node address.

1. Input the VALUE to be deleted
2. if ((START →DATA) is equal to **VALUE**) // if Data is in First Node
   (a) PTR = START
   (b) START = START →Next
   (c) Set free the node PTR, which is deleted
   (d) Exit
3. TEMP = START
4. while ((TEMP →Next →Next) not equal to NULL)) // if Data may be in the middle node
   (a) if ((TEMP →NEXT →DATA) equal to **VALUE**)
      (i) PTR = TEMP →Next
      (ii) TEMP →Next = PTR →Next
      (iii) Free (PTR) //Free the memory of the deleted node
      (iv) Exit
   (b) TEMP = TEMP →Next
5. if ((TEMP →next →DATA) == **VALUE**) // if Data is in LAST Node
   (a) PTR = TEMP →Next
   (b) Set free the node PTR, which is deleted
   (c) TEMP →Next = NULL
   (d) Exit
6. Display “VALUE not found”
7. Exit