6.7.2 Deleting Nodes

In this section, we present the algorithms to delete the nodes from the linked list. Deleting a node from the linked list has the following three instances.

1. Deleting the first node of the linked list.
2. Deleting the last node of the linked list.
3. Deleting the specified node within the linked list.

In order to delete a node from the list, it is necessary to search for location of deletion. The steps involved in deleting the node from the linked list are as follows :

1. If the linked list is empty then display the message "Deletion is not possible".
2. If the node to be deleted is the first node (pointed to by **head** pointer) then set the pointer **head** to point to the second node in the linked list.
3. If the node to be deleted is the last node, then go on locating the last but one node and set its link field to point to **null pointer.**
4. If the situation is other than the above three, then delete the node from the specified position within the linked list.

Let us develop algorithms for each of these instances.

6.7.2A Deleting the First Node

An algorithm for deleting the first node from the linked list is given below :

1. If the list is **not** empty then check whether the element belongs to the first node of the list.
2. Move the **start** pointer to the second node.
3. Free the first node.

Look at the Figure 6.9

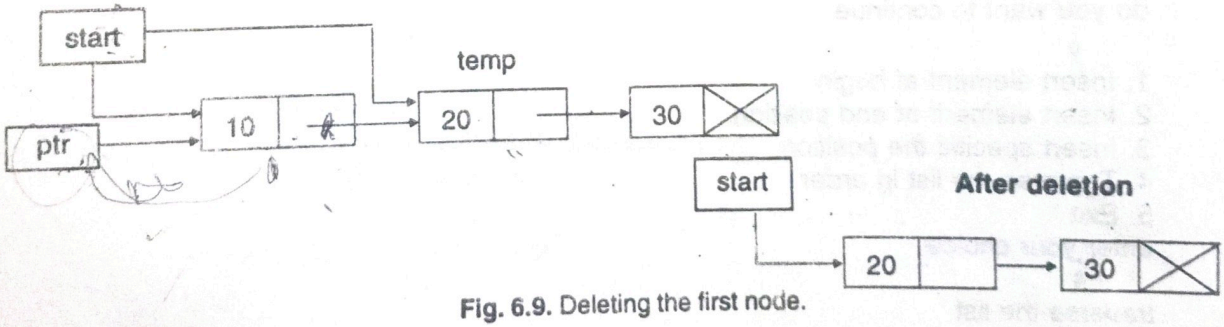

Fig. 6.9. Deleting the first node.

The C code for the above algorithm is :

```
void dele_beg (void)
{     NODE *ptr ;
      if (start = = NULL)
      { return ; }
      else
      {     ptr = start ;
            start = start → next ;
            free (ptr) ;
      }
}
```

6.7.2B Deleting the Last Node

An algorithm for deleting the last node from the linked list is given below :

1. If the linked list is not empty, go on traversing the list till the last node.
2. Set link field of the last but one node to NULL.
3. Free the last node.

Look at the Figure 6.10

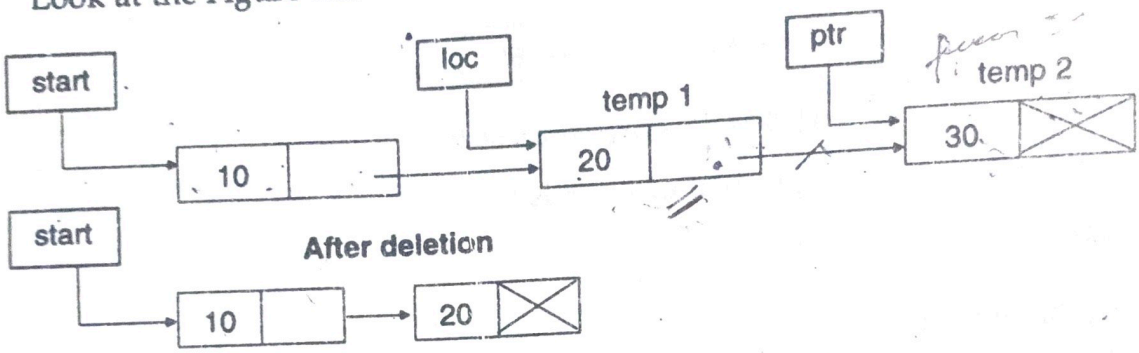

Fig. 6.10. Deleting the last node.

The C code for deleting from end position is :

```
void dele_end( )
{
    NODE *ptr, *loc ;
    if (start = = NULL)
    { return ; }
    else if ((start) → next = = NULL)
    {   ptr = start ;
        start = NULL ;
        free (ptr) ;
    }
    else
    {   loc = start ;
        ptr = (start) → next ;
        while (ptr → next = NULL)
        {   loc = ptr ;
            ptr = ptr → next ;
        }
        loc → next = NULL ;
        free (ptr) ;
    }
}
```

6.7.2C Deleting the Node from Specified Position

An algorithm for deleting the node in between two nodes Node A and Node B is given below :

1. If the linked list is not empty, then make the link field of Node A to point to Node B.
2. Free the node between Node A and Node B.

Look at the Figure 6.11

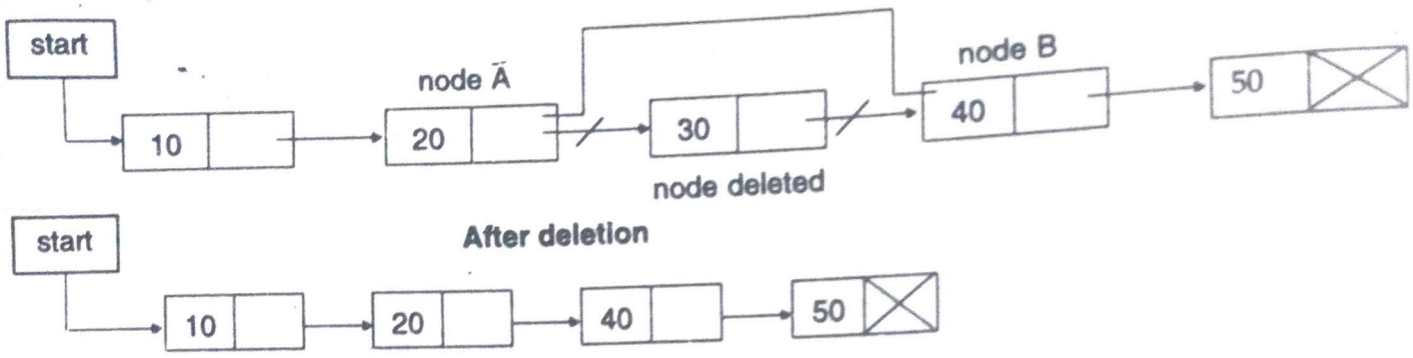

Fig. 6.11. Deleting the last node.

The C code for deleting from specified position is :

```
dele_spe( )
{
    NODE *ptr, *loc ;
    int temp ;
    printf ("Enter the which element do you want to delete\n") ;
    scanf ("%d", &temp) ;
    ptr = start ;
    if (start = = NULL)
    {
        printf ("Empty list\n") ;
        return ;
    }
    else
    {
        loc = ptr ;
        while (ptr != NULL)
        {
            if (ptr → info = = temp)
            {
                loc → next = ptr → next ;
                free (ptr) ;
                return ;
            }
            loc = ptr ;
            ptr = ptr → next ;
        }
    }
}
```