Data Structure

Linked Lists

Instructors:
Md Nazrul Islam Mondal &
Rizoan Toufiq
Department of Computer Science & Engineering
Rajshahi University of Engineering &
Technology Rajshahi-6204

Outline

- Introduction
- Linked List
- Representation of Linked Lists in Memory
- Traversing a Linked List
- Searching a Linked List
- Memory Allocation; Garbage Collection
- Insertion into a Linked List
- **Deletion from a Linked List**
- Header Linked List
- Two Way Lists

Deletion from a Linked List

Deletion from a Linked List

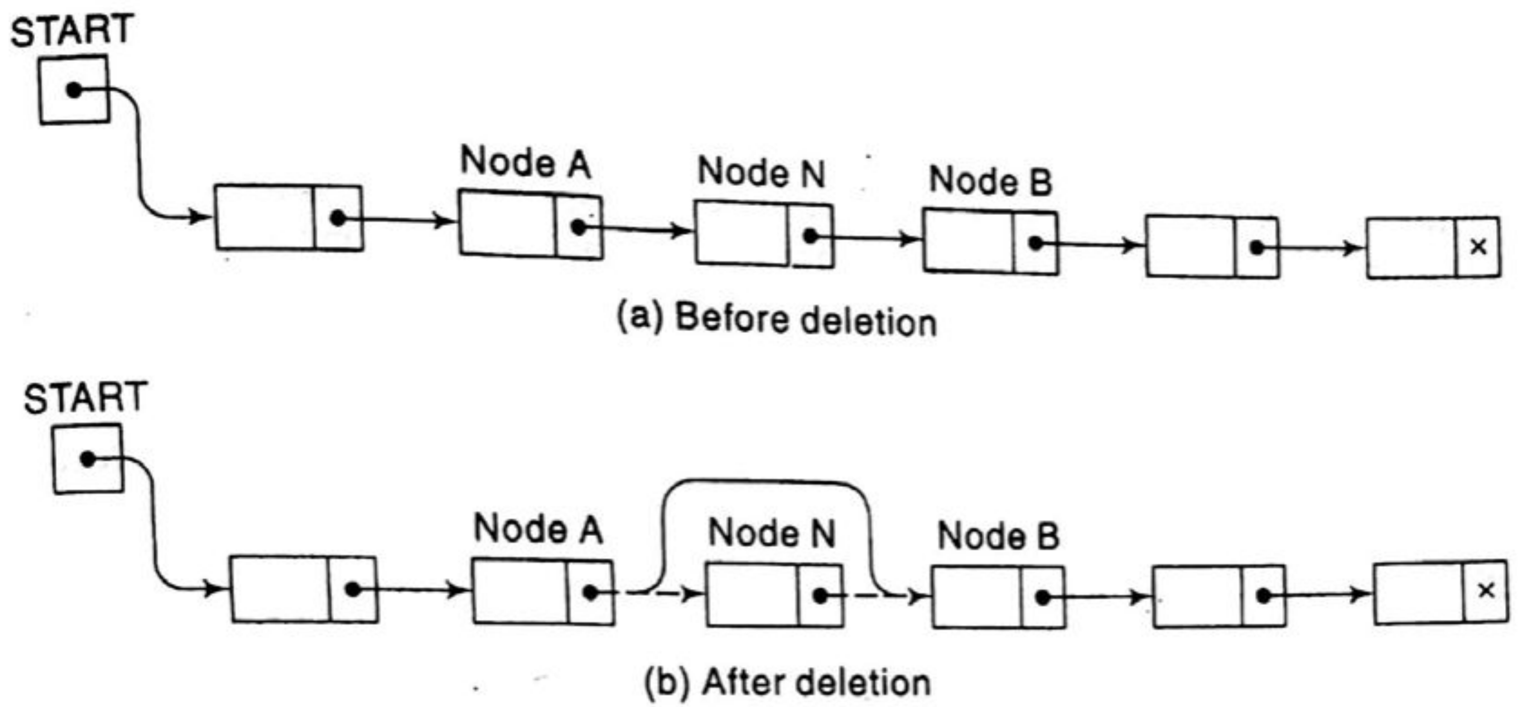


Fig. 5.22

Deletion from a Linked List

- Three Pointer fields are changed as follows:

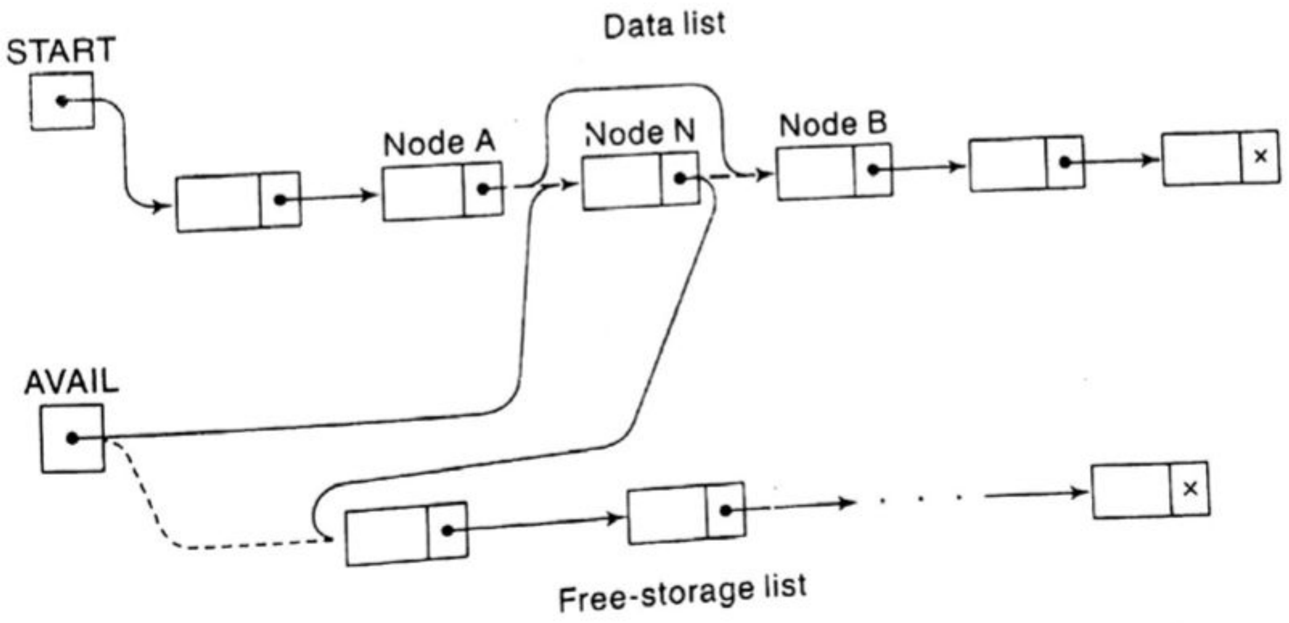


Fig. 5.23

Deletion from a Linked List

- Add deleted node with **available list**

LINK[LOC]←AVAIL

AVAIL ← LOC

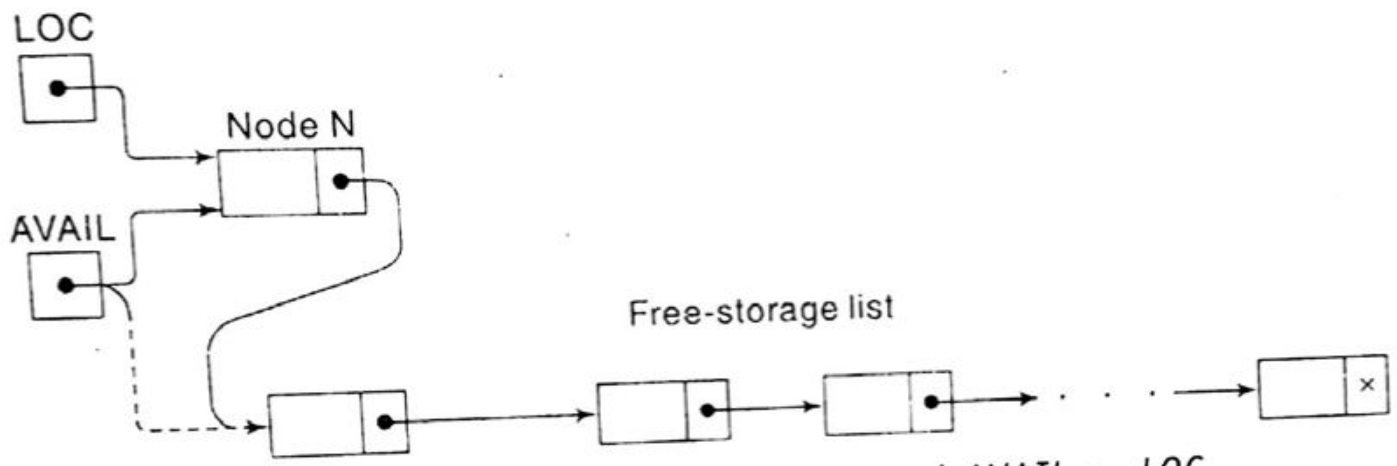
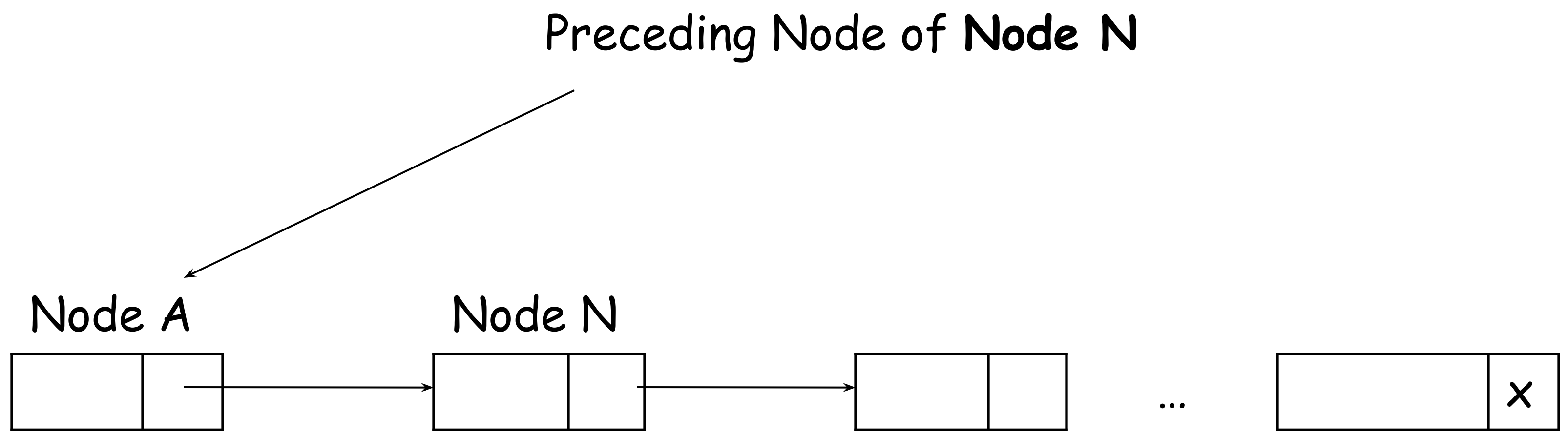


Fig. 5.25 *LINK[LOC] := AVAIL and AVAIL := LOC*

Deletion from a Linked List



Deletion from a Linked List (Deleting the Node following a Given Node)

- **Case 1:** Delete first node

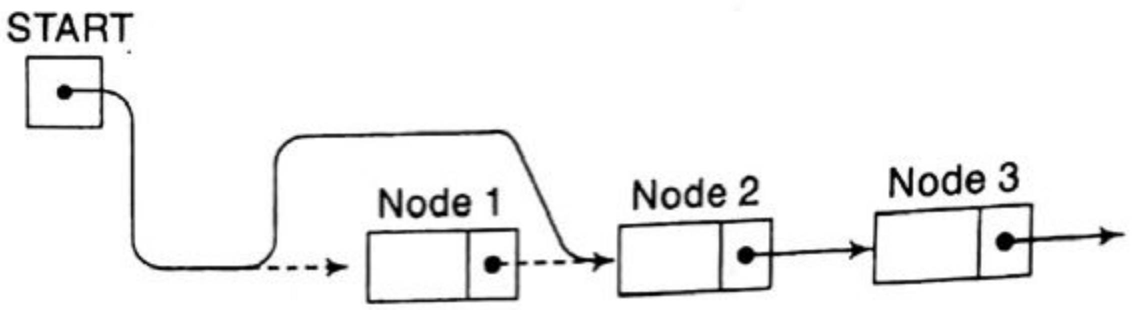


Fig. 5.26 *START := LINK[START]*

START←LINK[START]

Deletion from a Linked List
(Deleting the Node following a Given Node)

- **Case 2:** Delete other node

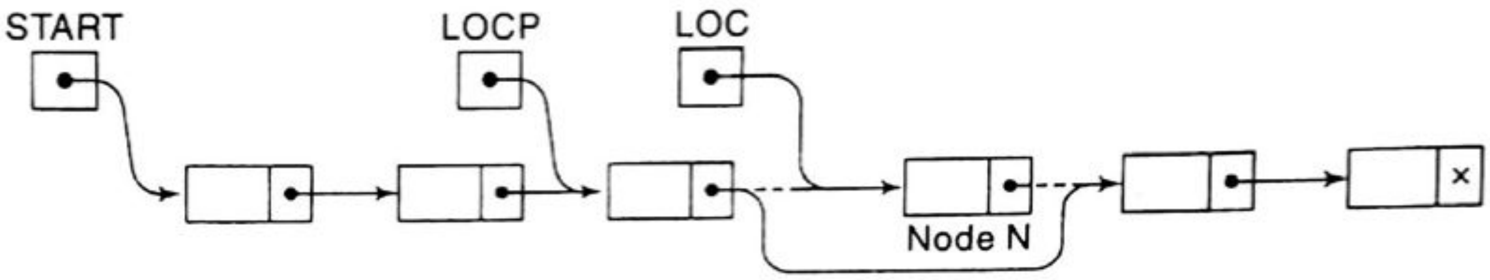
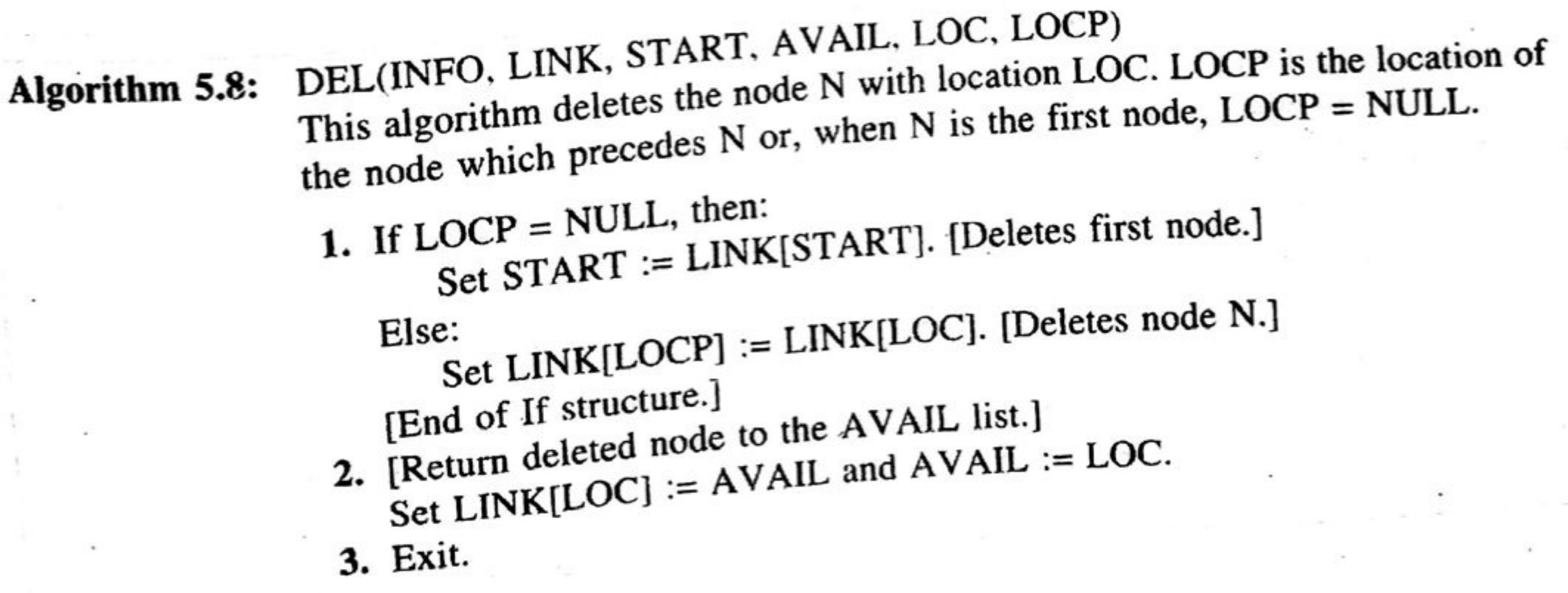


Fig. 5.27 *LINK[LOCP] := LINK[LOC]*

LINK[LOCP]←LINK[LOC]

Deletion from a Linked List
(Deleting the Node following a Given Node)



Algorithm 5.8: DEL(INFO, LINK, START, AVAIL, LOC, LOCP)
This algorithm deletes the node N with location LOC. LOCP is the location of the node which precedes N or, when N is the first node, LOCP = NULL.

```
1. If LOCP = NULL, then:
        Set START := LINK[START]. [Deletes first node.]
   Else:
        Set LINK[LOCP] := LINK[LOC]. [Deletes node N.]
   [End of If structure.]
2. [Return deleted node to the AVAIL list.]
   Set LINK[LOC] := AVAIL and AVAIL := LOC.
3. Exit.
```

Deletion from a Linked List
(Deleting the Node with a Given ITEM of Information)

Step1: Search the item i.e. FINDB()

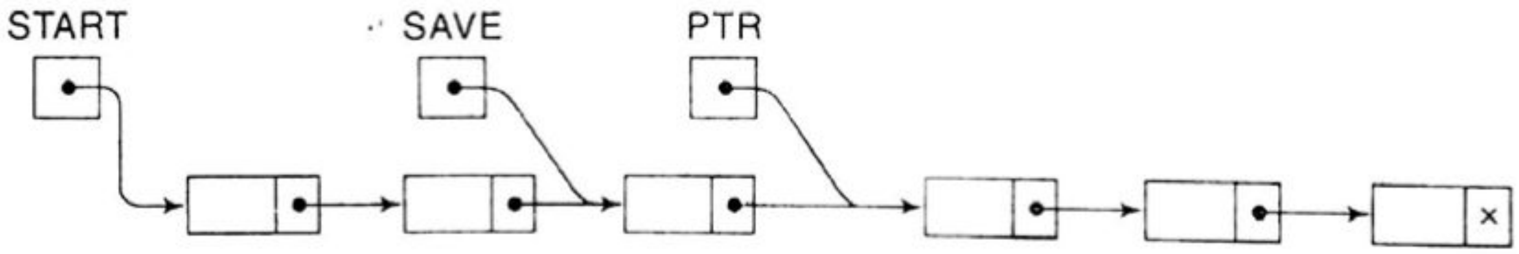


Fig. 5.20

```
SAVE:=PTR
PTR:=LINK[PTR]
```

Deletion from a Linked List
(Deleting the Node with a Given ITEM of Information)

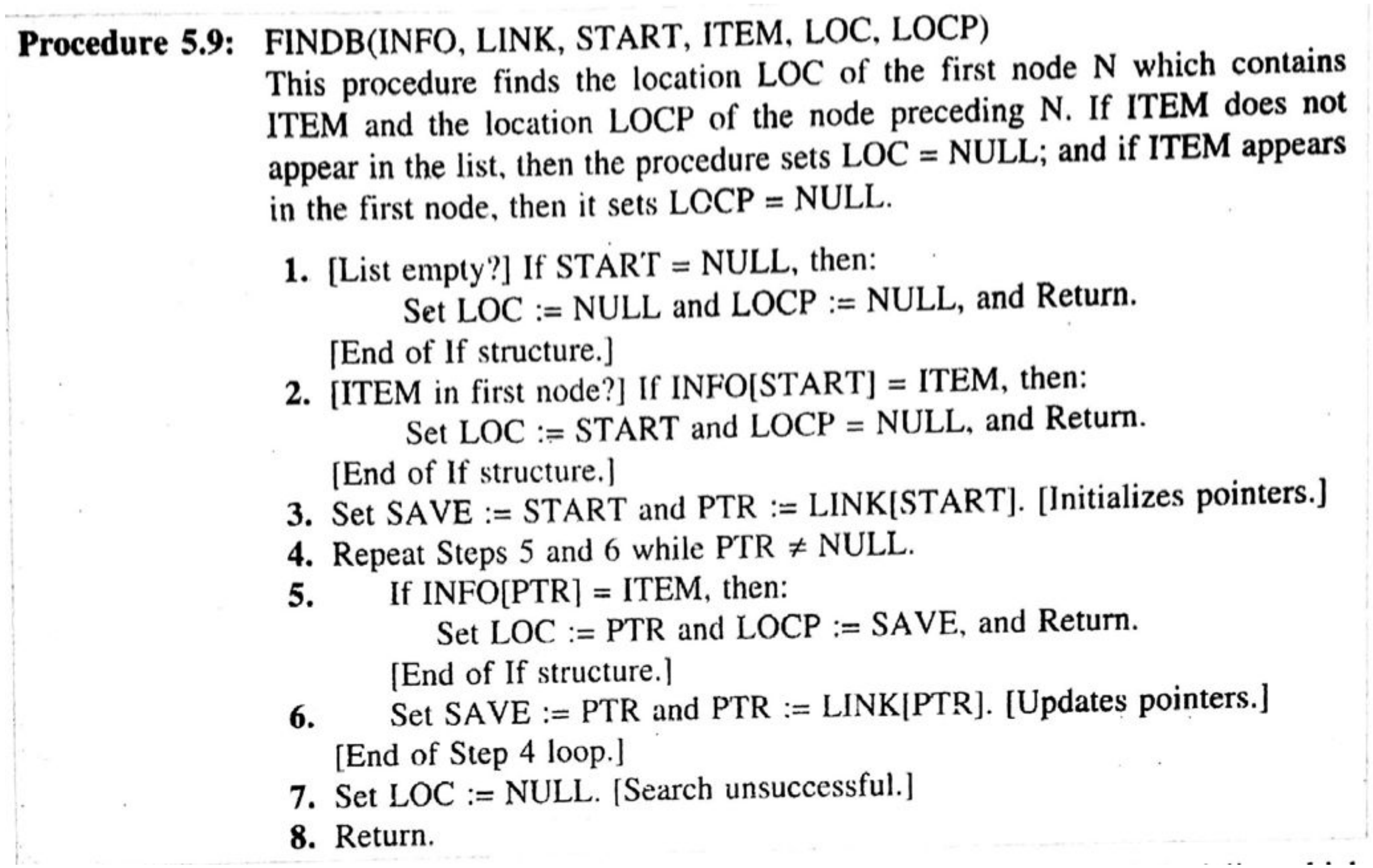
Output of the searching algorithm:

Case – 1: LIST is NULL, LOC:=NULL,LOCP:=NULL
Case – 2: ITEM in FIRST Node, LOC:=START, LOCP:=NULL
Case – 3: ITEM in Other Node, LOC:=PTR, LOCP:=SAVE
Case – 4: ITEM is not in LIST, LOC:=NULL, LOCP:=NULL

Step2: Perform Delete Operation i.e. DELETE()

Deletion from a Linked List

(Deleting the Node with a Given ITEM of Information)

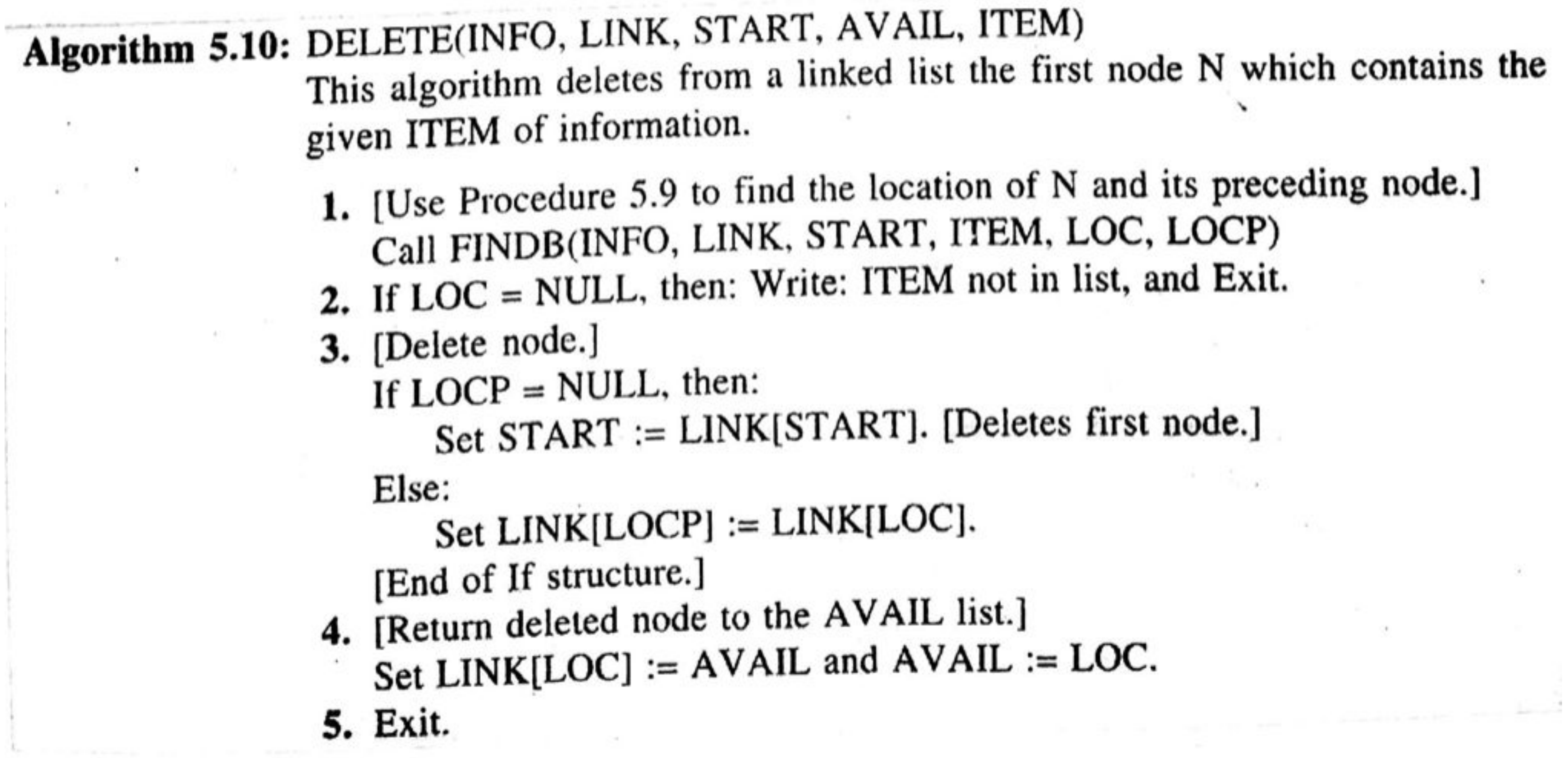


Procedure 5.9: FINDB(INFO, LINK, START, ITEM, LOC, LOCP)

This procedure finds the location LOC of the first node N which contains ITEM and the location LOCP of the node preceding N. If ITEM does not appear in the list, then the procedure sets LOC = NULL; and if ITEM appears in the first node, then it sets LOCP = NULL.

```
1. [List empty?] If START = NULL, then:
           Set LOC := NULL and LOCP := NULL, and Return.
   [End of If structure.]
2. [ITEM in first node?] If INFO[START] = ITEM, then:
           Set LOC := START and LOCP = NULL, and Return.
   [End of If structure.]
3. Set SAVE := START and PTR := LINK[START]. [Initializes pointers.]
4. Repeat Steps 5 and 6 while PTR ≠ NULL.
5.     If INFO[PTR] = ITEM, then:
               Set LOC := PTR and LOCP := SAVE, and Return.
       [End of If structure.]
6.     Set SAVE := PTR and PTR := LINK[PTR]. [Updates pointers.]
   [End of Step 4 loop.]
7. Set LOC := NULL. [Search unsuccessful.]
8. Return.
```

Deletion from a Linked List
(Deleting the Node with a Given ITEM of Information)



Algorithm 5.10: DELETE(INFO, LINK, START, AVAIL, ITEM)
This algorithm deletes from a linked list the first node N which contains the given ITEM of information.

```
1. [Use Procedure 5.9 to find the location of N and its preceding node.]
   Call FINDB(INFO, LINK, START, ITEM, LOC, LOCP)
2. If LOC = NULL, then: Write: ITEM not in list, and Exit.
3. [Delete node.]
   If LOCP = NULL, then:
       Set START := LINK[START]. [Deletes first node.]
   Else:
       Set LINK[LOCP] := LINK[LOC].
   [End of If structure.]
4. [Return deleted node to the AVAIL list.]
   Set LINK[LOC] := AVAIL and AVAIL := LOC.
5. Exit.
```

Deletion from a Linked List

Source Code

END