Lecture 10
Linked list Deletion and more

Prepared By:

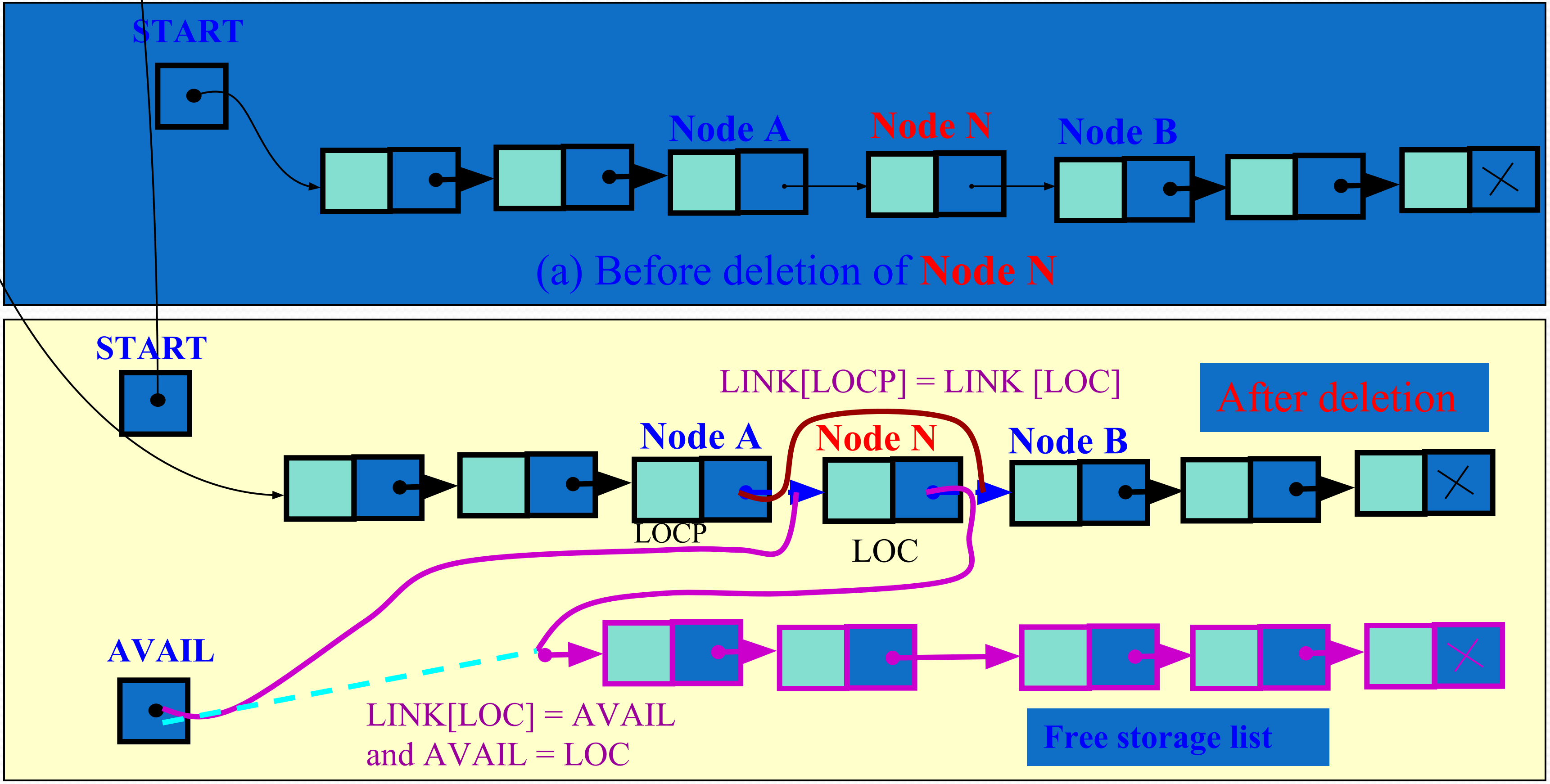
Afsana Begum
Lecturer,
Software Engineering Department,
Daffodil International University.

Contents

- Algorithm of Deletion
- **circular linked list and its advantage and examples**
- Linear linked list vs Circular Linked Lists
- Array versus Linked Lists

Deletion from a Linked List

- The next pointer field of node-A now points to node-B, where node-N previously pointed
- The next pointer field of N now points to the original first node in the free pool, where AVAIL previously pointed.
- AVAIL now points to the deleted node-N



(a) Before deletion of Node N

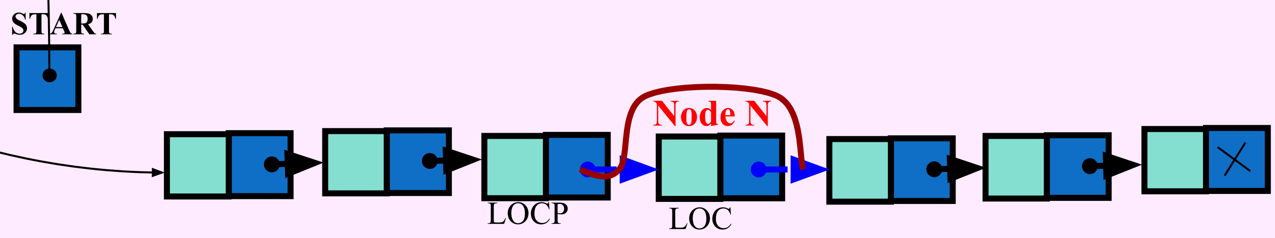
LINK[LOC] = AVAIL
and AVAIL = LOC

Finding a Node and Delete

This procedure finds the location N[LINK] of the first node N which contains.

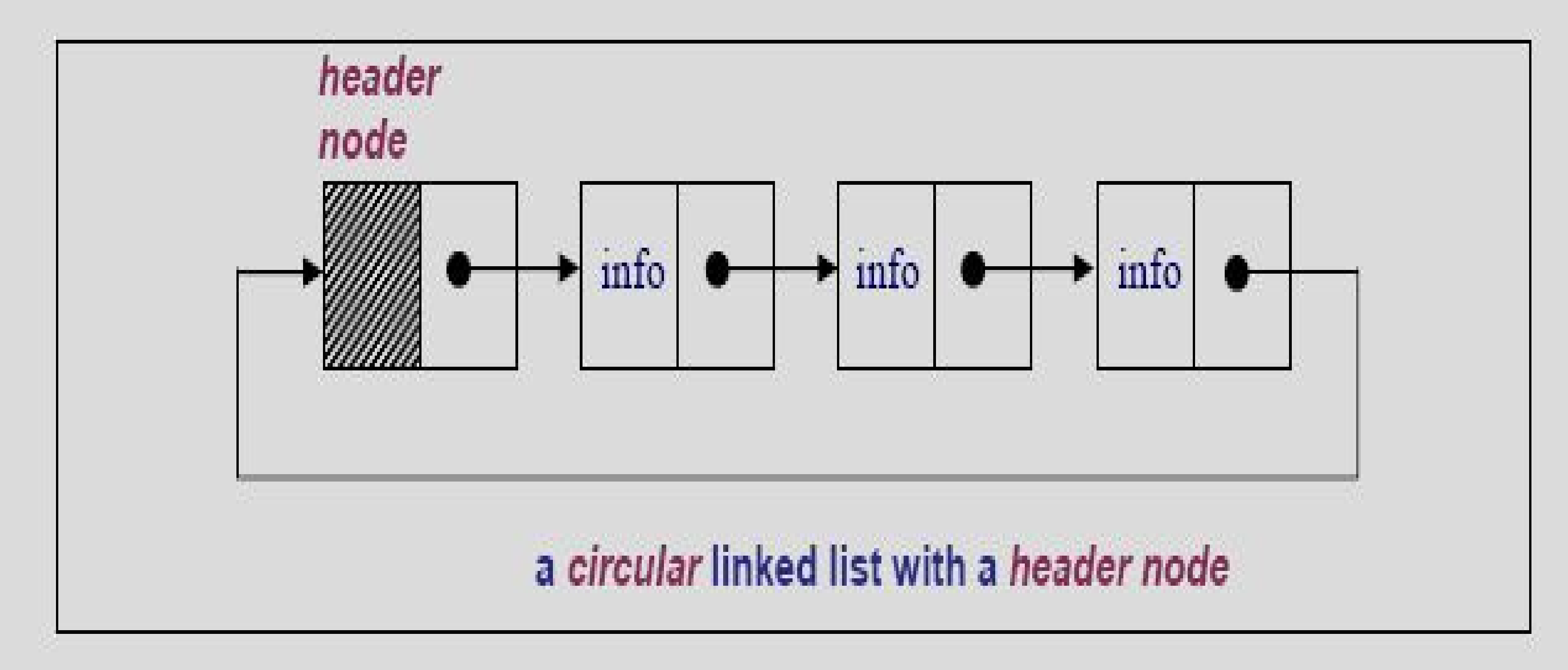
```
1.  [List Empty?] If START = NULL then:
     Write UNDERFLOW and return.
2.  [ITEM in the first node?] If N[DATA ] = ITEM and N[LINK]=NULL then:
     Set START=NULL and N[LINK] = NULL and N[DATA]=null and return.

3.  Repeat step 4 until N[LINK] = NULL
4.  If  N[DATA] = ITEM and N[LINK]!=NULL then:
     Set PRE_N[LINK] =N[LINK] and N[LINK] = NULL and N[DATA]=null and Return.
     Else
     Write Not found and return.
5.  Exit.
```



Variations of Linked Lists

- *Circular linked lists*
  - The last node points to the first node of the list
  - How do we know when we have finished traversing the list? (Tip: check if the pointer of the current node is equal to the head.)



Variations of Linked Lists

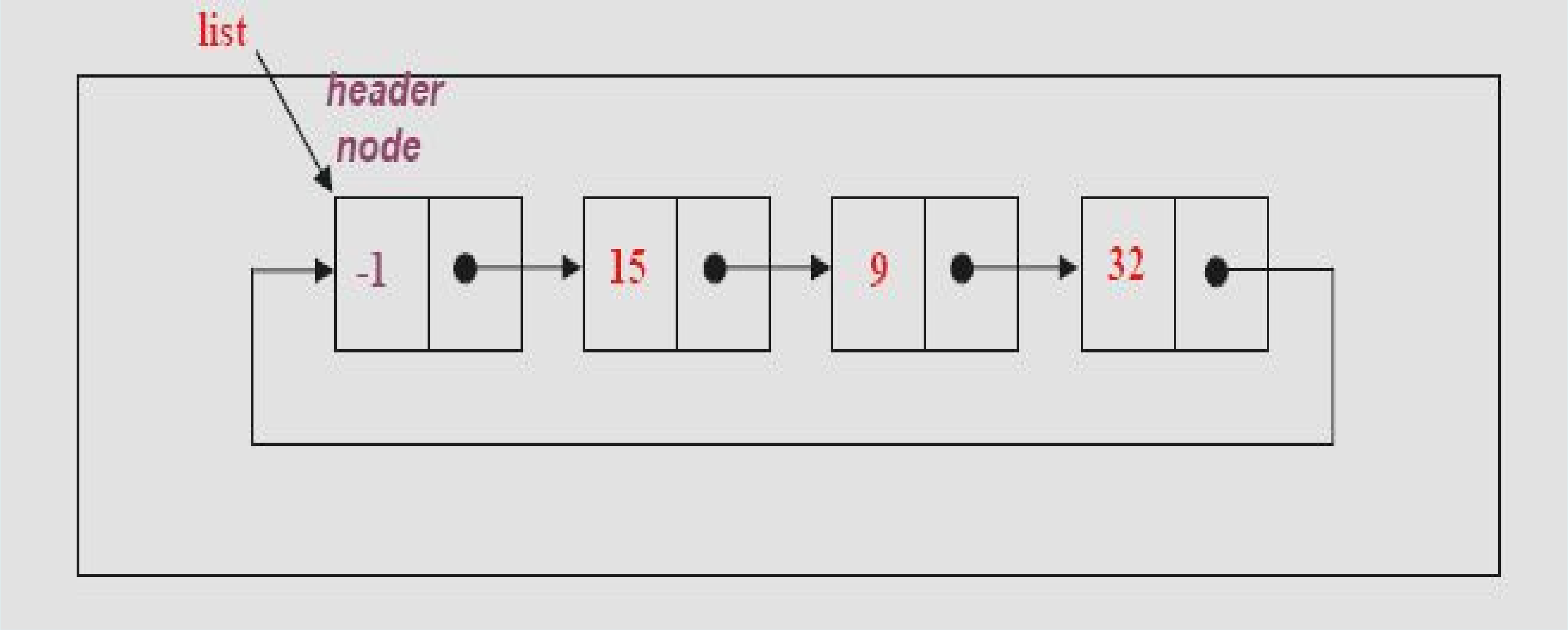
circular linked list:

- In a circular linked list there are two methods to know whether a node is the first node or not.

  - Either a external pointer, ***list***, points the first node or
  - A ***header node*** is placed as the first node of the circular list.

- The header node can be separated from the others by either heaving a ***sentinel value*** as the info part or
- having a dedicated ***flag*** variable to specify if the node is a header node or not.

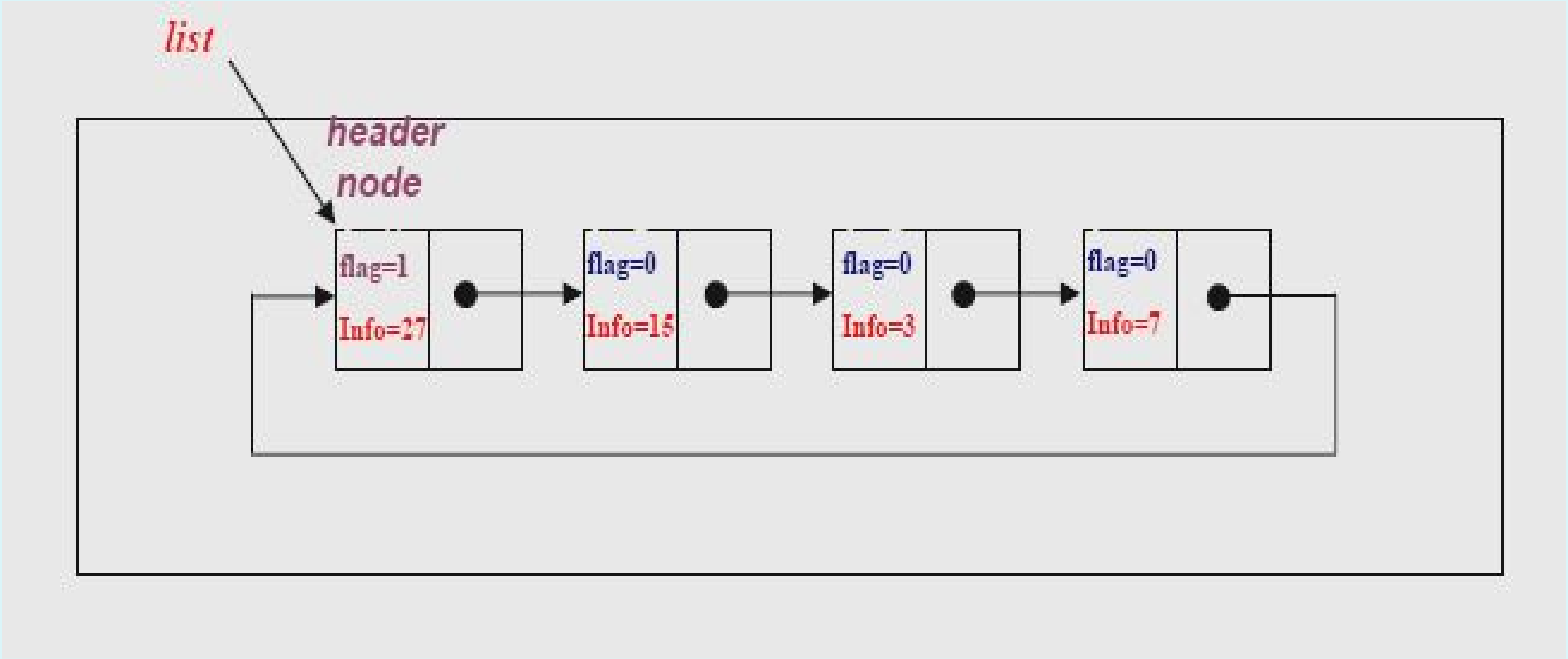
CIRCULAR LIST with header node



•The header node in a circular list can be specified by a ***sentinel value*** or a dedicated ***flag***:

•***Header Node with Sentinel:*** Assume that info part contains positive integers. Therefore the info part of a header node can be -1.

CIRCULAR LIST with header node



- ***Header Node with Flag:*** In this case a extra variable called flag can be used to represent the header node.
- For example flag in the header node can be 1, where the flag is 0 for the other nodes.

Example of application of circular linked list

- A good example of an application where circular linked list should be used is a timesharing problem solved by the operating system.
- In a timesharing environment, the operating system must maintain a list of present users and must alternately allow each user to use a small slice of CPU time, one user at a time.
- The operating system will pick a user, let him/her use a small amount of CPU time and then move on to the next user, etc.
- For this application, there should be no NIL pointers unless there is absolutely no one requesting CPU time.

Advantages

- Each node is accessible from any node.
- **Disadvantage:**
- Danger of an infinite loop !

Linear linked list vs Circular Linked Lists

- In **linear linked** lists if a list is traversed (all the elements visited) an external pointer to the list must be preserved in order to be able to reference the list again.
- **Circular linked** lists can be used to help the traverse the same list again and again if needed. A circular list is very similar to the linear list where in the circular list the pointer of the last node points not NULL but the first node.

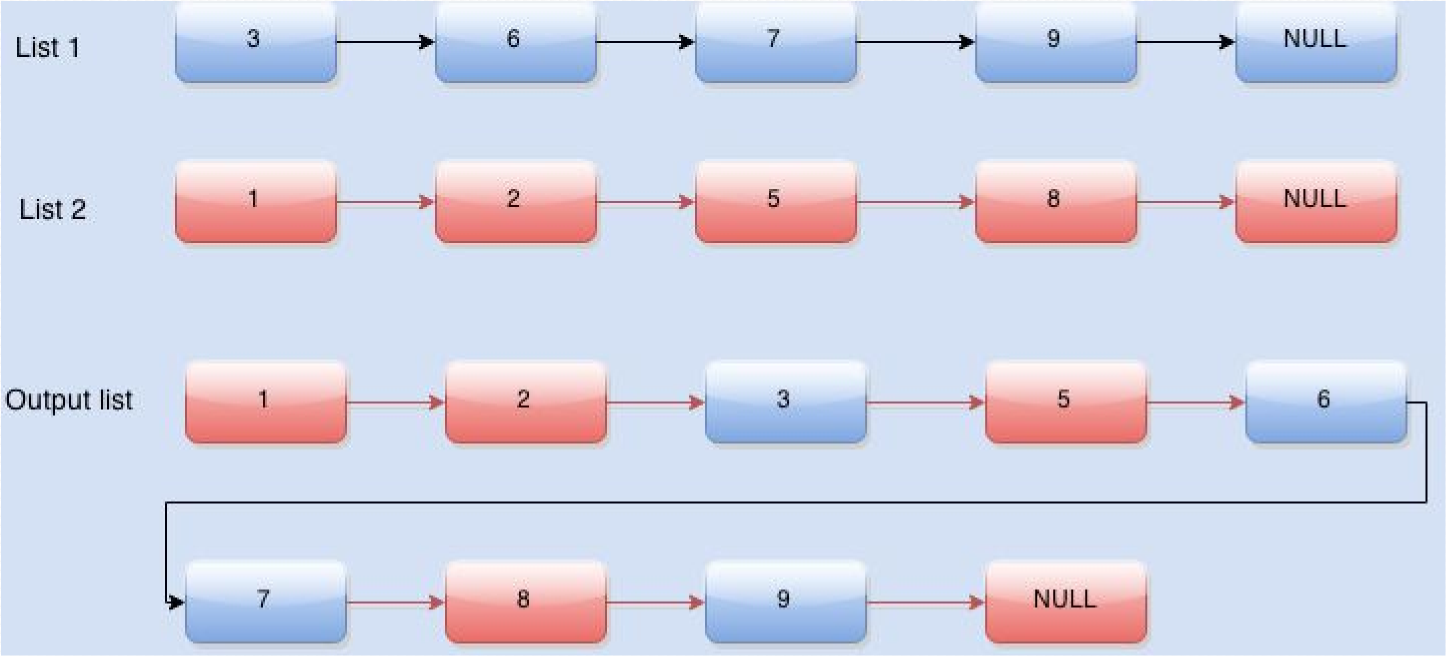
Array versus Linked Lists

- **Linked lists** are more complex to code and manage than arrays, but they have some distinct advantages.

- **Dynamic:** a linked list can easily grow and shrink in size.
  - We don't need to know how many nodes will be in the list. They are created in memory as needed.
  - In contrast, the size of a C++ array is fixed at compilation time.

- **Easy and fast insertions and deletions**
  - To insert or delete an element in an array, we need to copy to temporary variables to make room for new elements or close the gap caused by deleted elements.
  - With a linked list, no need to move other nodes. Only need to reset some pointers.

Link List Merge



Question?