# CSC 224: Data Structures

Masinde Muliro University



By: Omieno Kelvin

# LISTS

- For stacks and queues the basic operations are Insertions, and deletion- which are restricted to the ends of the structures.
- No such limitations are imposed on the general lists as items may be inserted and /or deleted at any point in the list.

ADT List

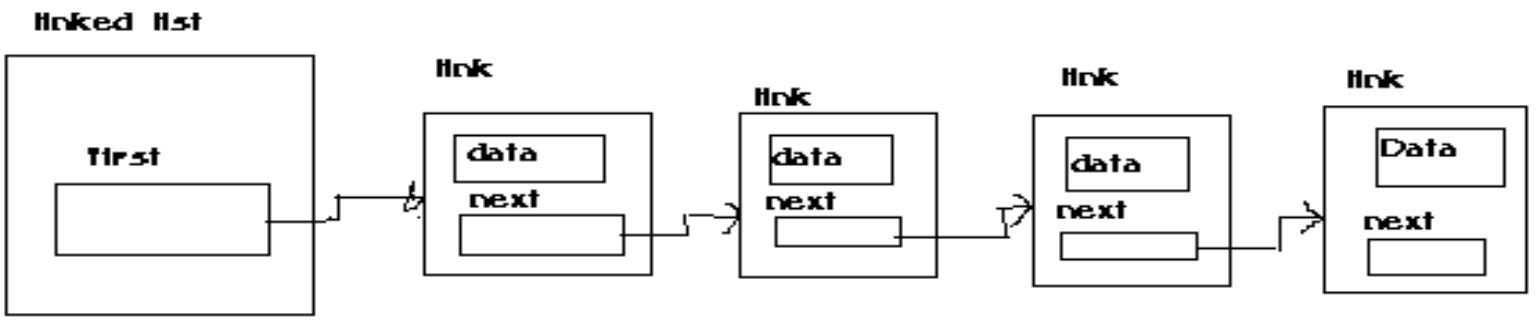
- Collection of data elements : a finite sequence (ordered set) of data items.

# Basic operations

- Construction:- create an empty list
- Empty:-check if the list is empty
- Traverse:-Go through the list or part of it, assessing and processing the elements in order.
- Insert:- add an item at any point in the list
- Delete:-Remove an item from the list at any point in the list

# Linked list

- An ordered collection of elements called nodes each of which has two parts:
- A data part that stores an element of the list
- A next part that stores a link or pointer that indicates the location of the node containing the next list element. If there is no next element, then special null value is used.



# Implementing the basic List operations

- Construction: to construct an empty list simply make first a Null Link, to indicate that it does not point to any node. First=null
- Empty: Check whether first=null
- Traverse: we begin by initializing some auxiliary pointer or reference; to point to the first node and we process the list elements stored in this node.
- Ref::initialize ref to first

- process the data part of the node pointed to by ref.

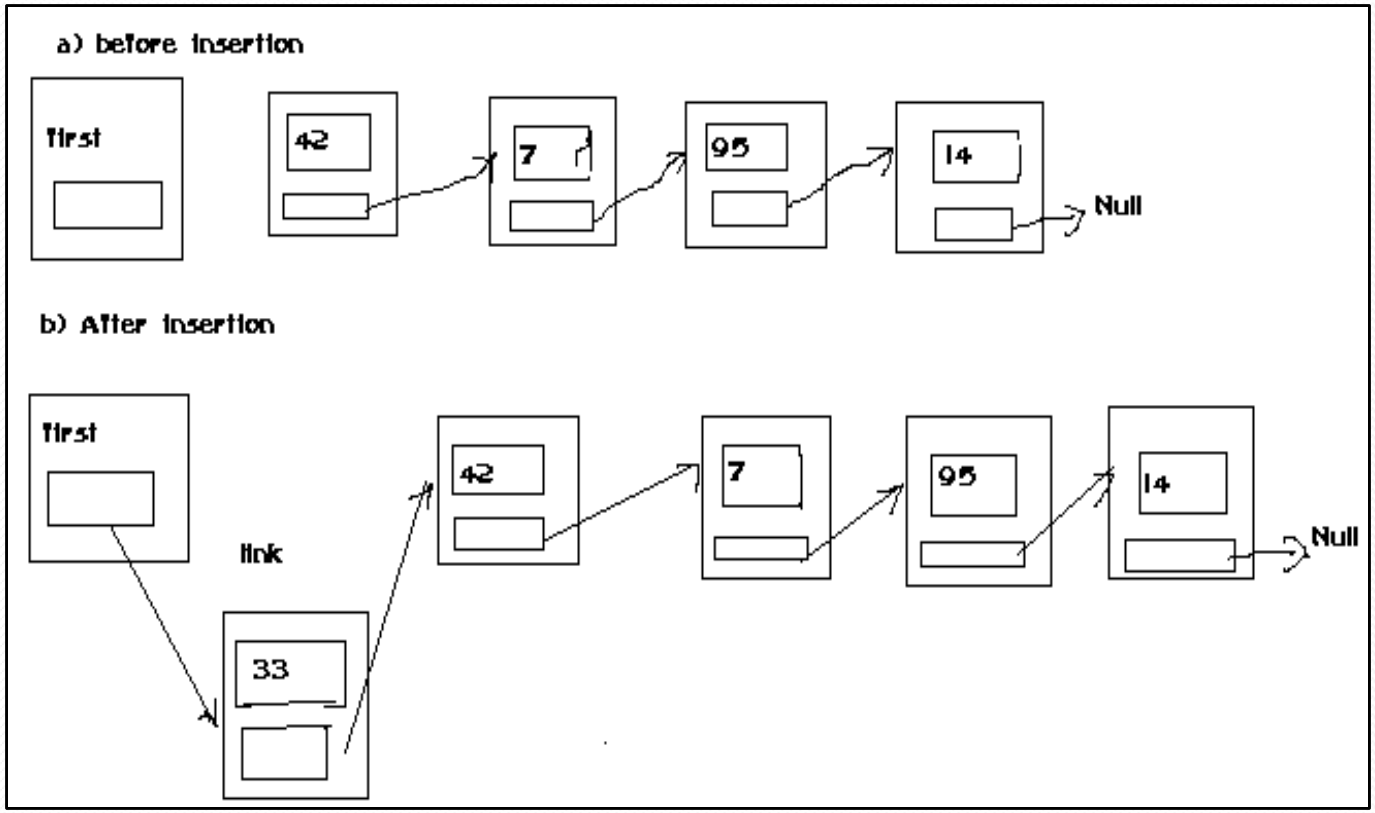
- To move to the next node, we set ref=link in the node pointed to by reference.

- We continue in this way until we reach the node having next=null signaling the end of the list.
- ref=first;

  while(ref!=null)

  { process data part of the node pointed to by ref;

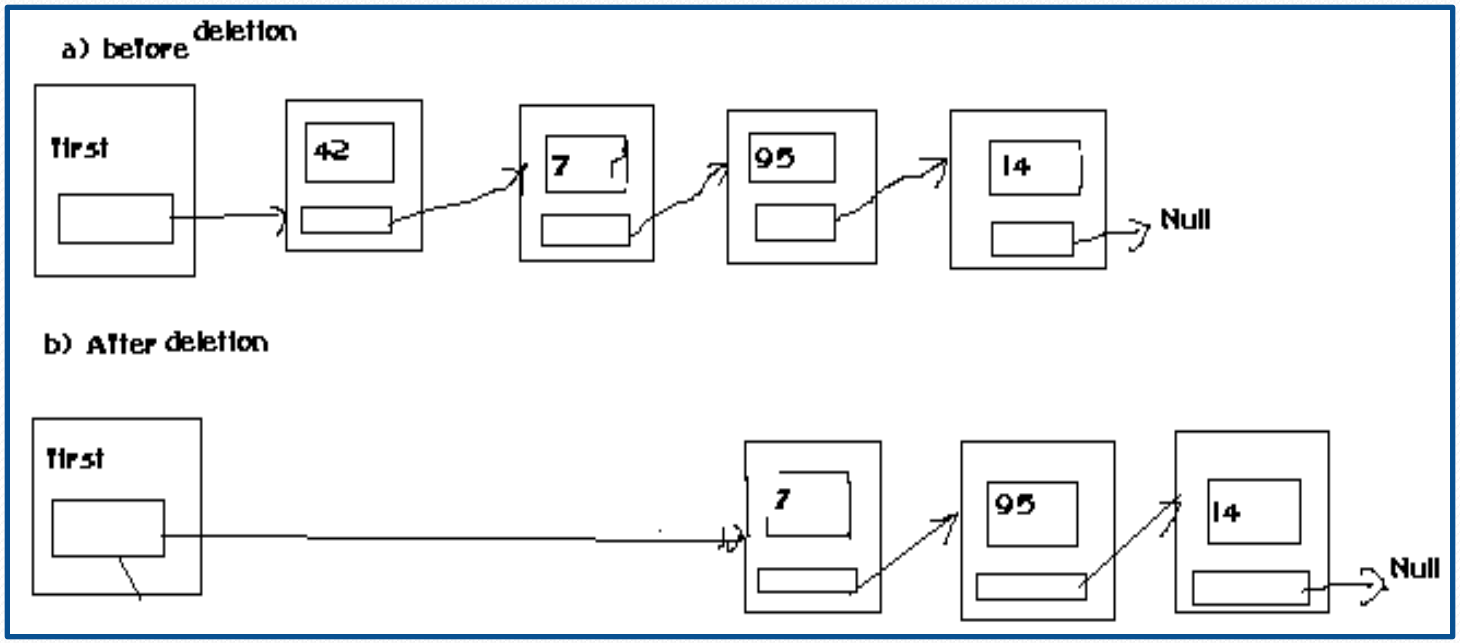
  ref=next part of node pointed to by ref.}
- Insertion: to insert a new data item in to a linked list, we must first obtain a new node and store the value in it's data part.
- NB: you not need to have a contiguous memory spaces you can store nodes any where unlike in Arrays.




- Adding at the end- create  a new node with data.
- Make the null part of the last node point to data part of the new created node and data  part point to the null pointer of the new node.

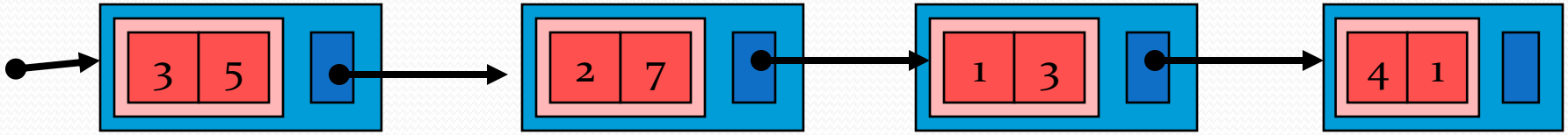
# deletion

- Deleting an element that has a predecessor
- Set the next part of the node pointed to by the pred. equal to the next part of the node pointed to by the ref.
- Return node pointed to by ref.

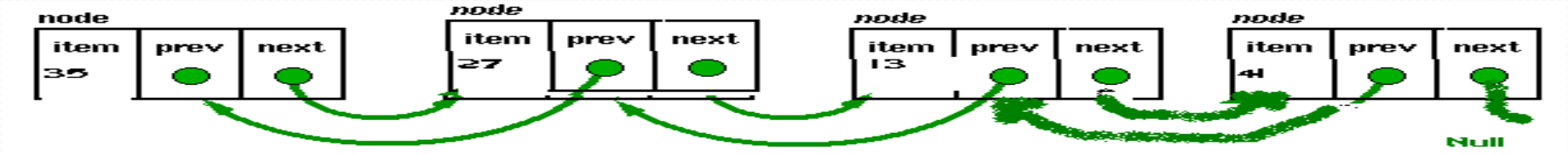


# Types of linked lists:

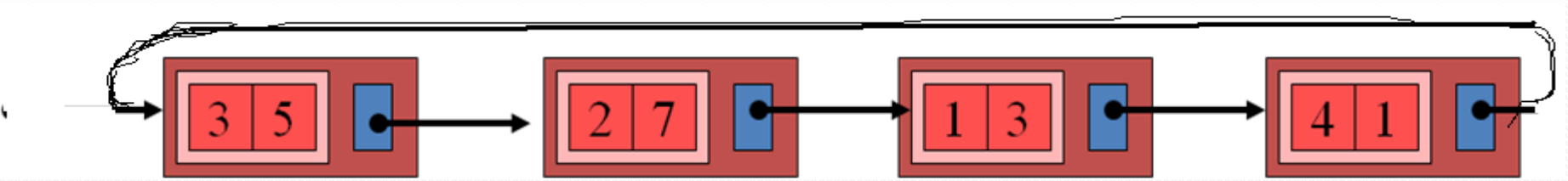
- Singly linked list-one direction of pointing




- Doubly linked lists- 2 directions of pointing




- Circular linked list



# Revision Questions

Q1. Describe how a {stack, queue} can be represented using a linked list? What is the main advantage of this representation over arrays?

Q2. Define the following terms:

i) Ordered List

ii) Homogenous ordered List

iii) Heterogeneous ordered list

Q3. Write a program that implements a stack using linked list ADT.

# END

Thank you.