ZAMAN UNIVERSITY

Data Structures and Algorithms

Chapter 2

Abstract Data Types

Outline

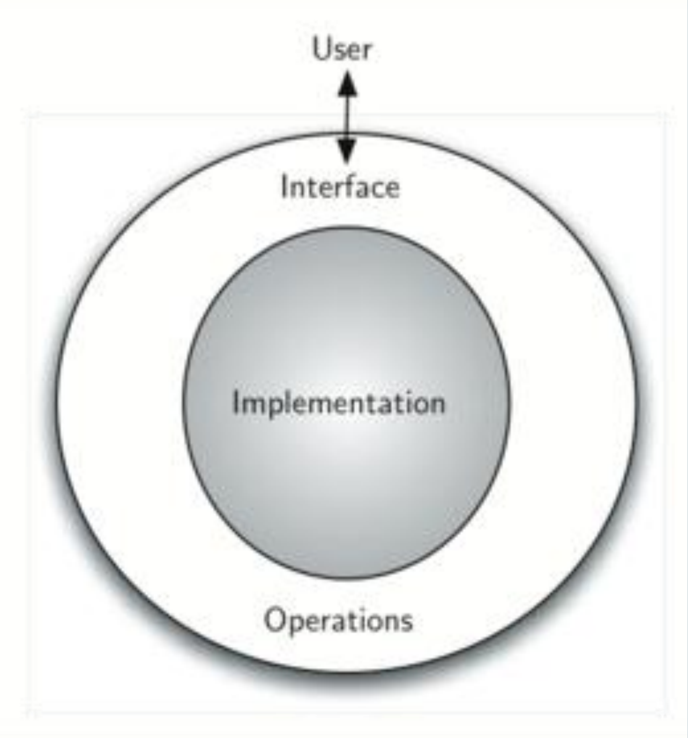
- Stacks
- Queues and Priority Queues
- Linked Lists
- Abstract Data Types
- Specialized Lists

Outline

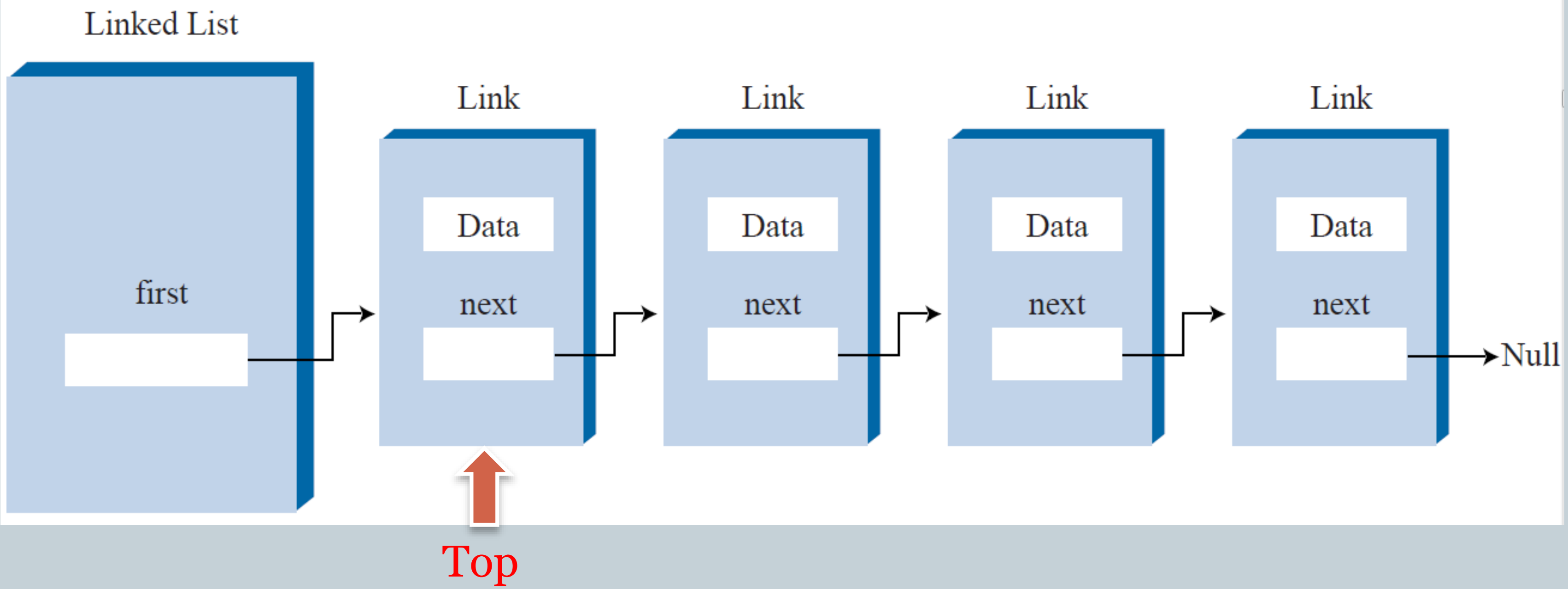
- Stacks
- Queues and Priority Queues
- Linked Lists
- Abstract Data Types
- Specialized Lists

Abstract Data Types

- What is Abstract Data Types? **ADT**, is a logical description of how we view the data and the operations, that are allowed without regard to how they will be implemented.
- Actually, Stack & Queue were implemented by using array in the previous lectures. Now, we will show how to implement Stack & Queue by using Linked List.
- Since, we will provide the same operations/interface to users.



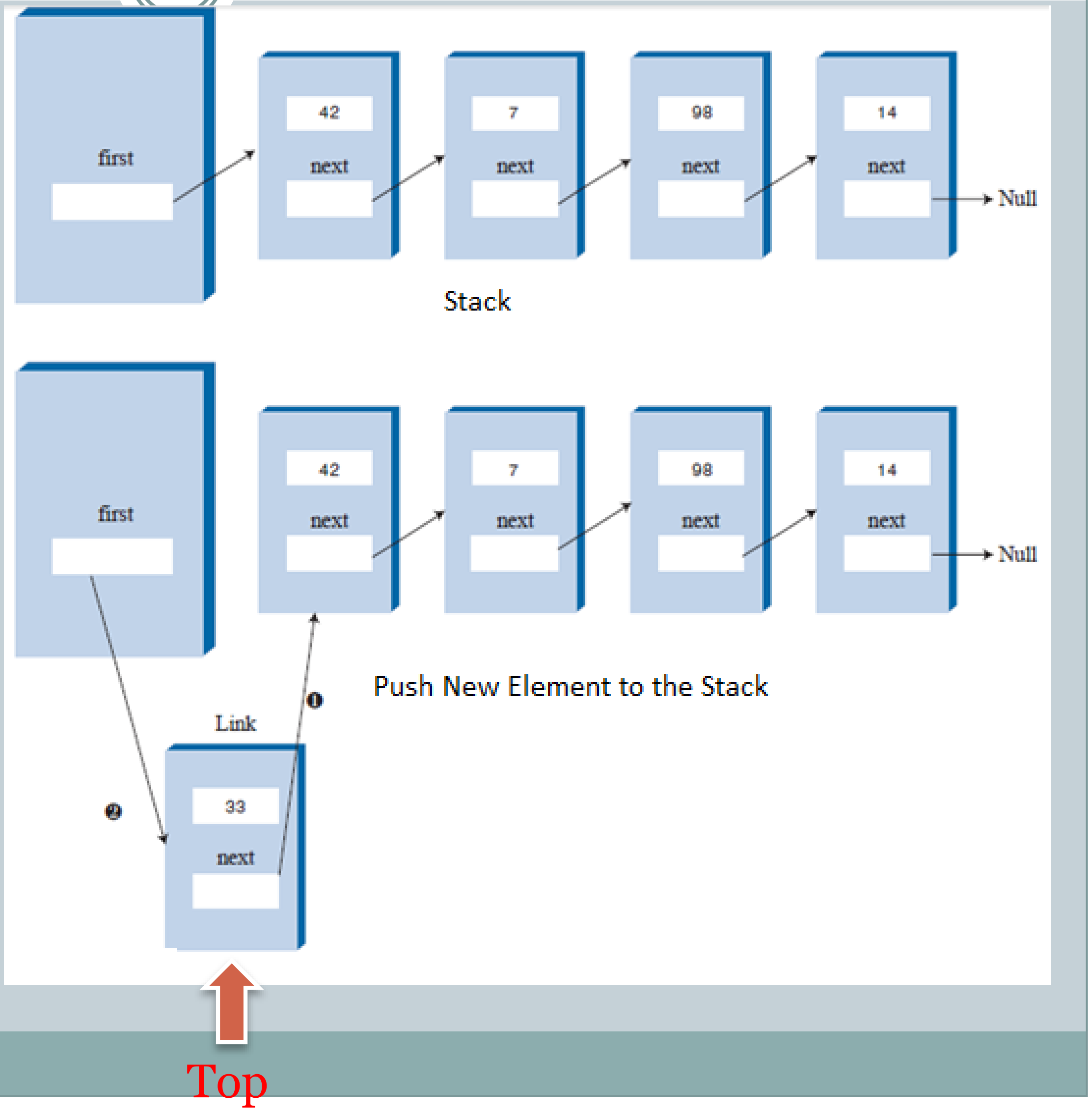
Implement a Stack using Linked List



Stack Operation (with Linked List): Push

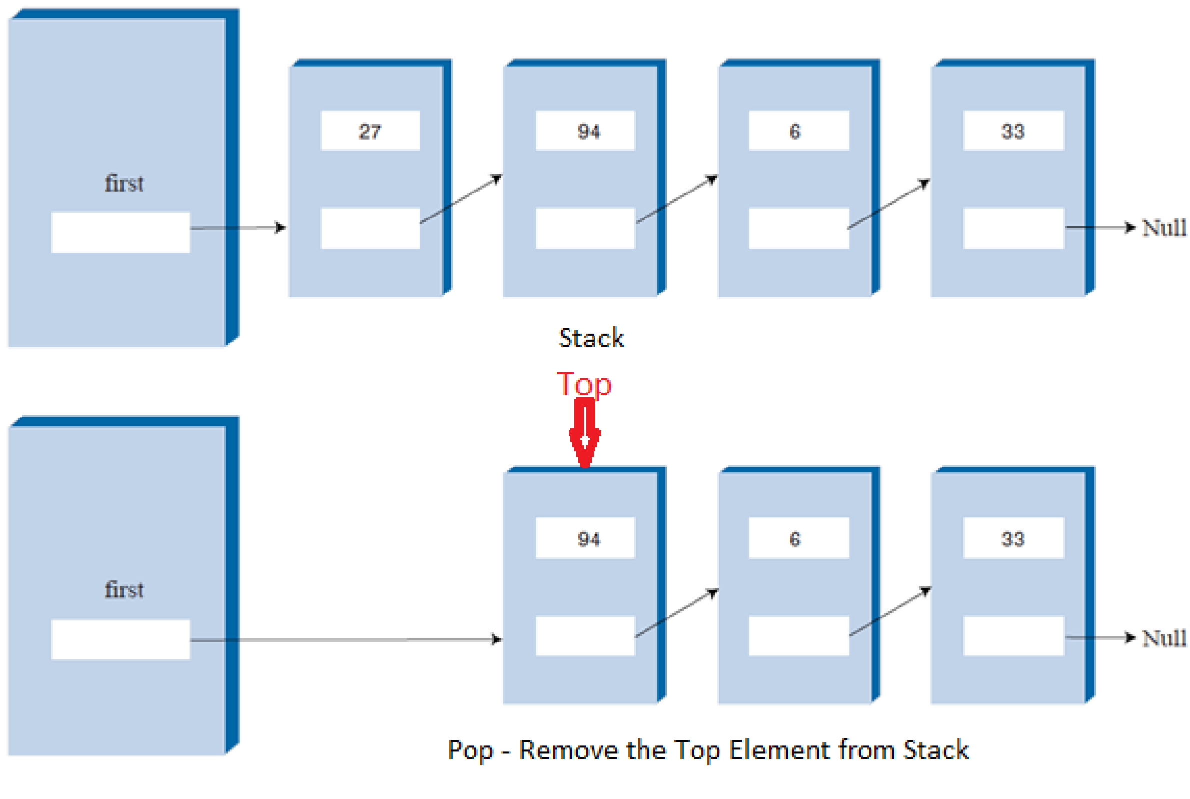
- Operation ***Push***

- Actually, it is the operation of insertion new link to the first

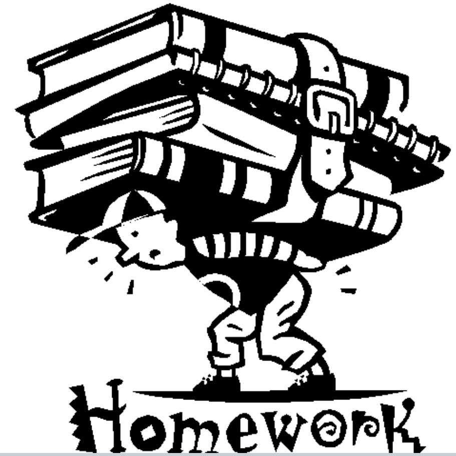


Stack Operation (with Linked List): Pop

- Operation ***Pop***
- Remove the first Link from Linked List



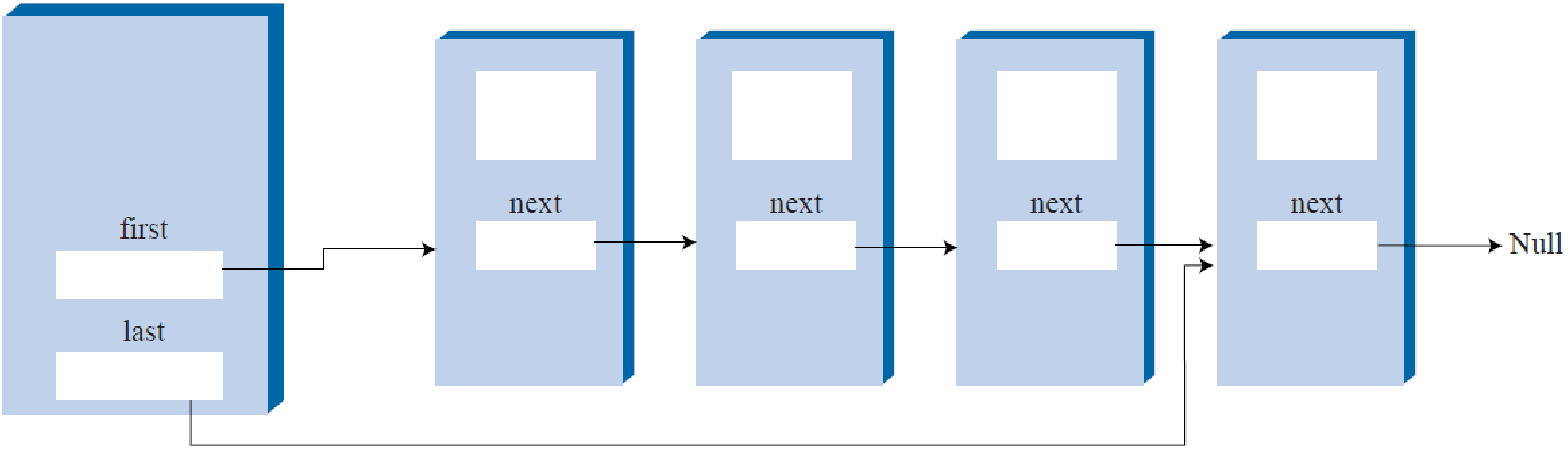
Pop - Remove the Top Element from Stack



- Create Stack by Using Linked List with the following operations: Push, Pop, Peek, Size, isEmpty, isFull

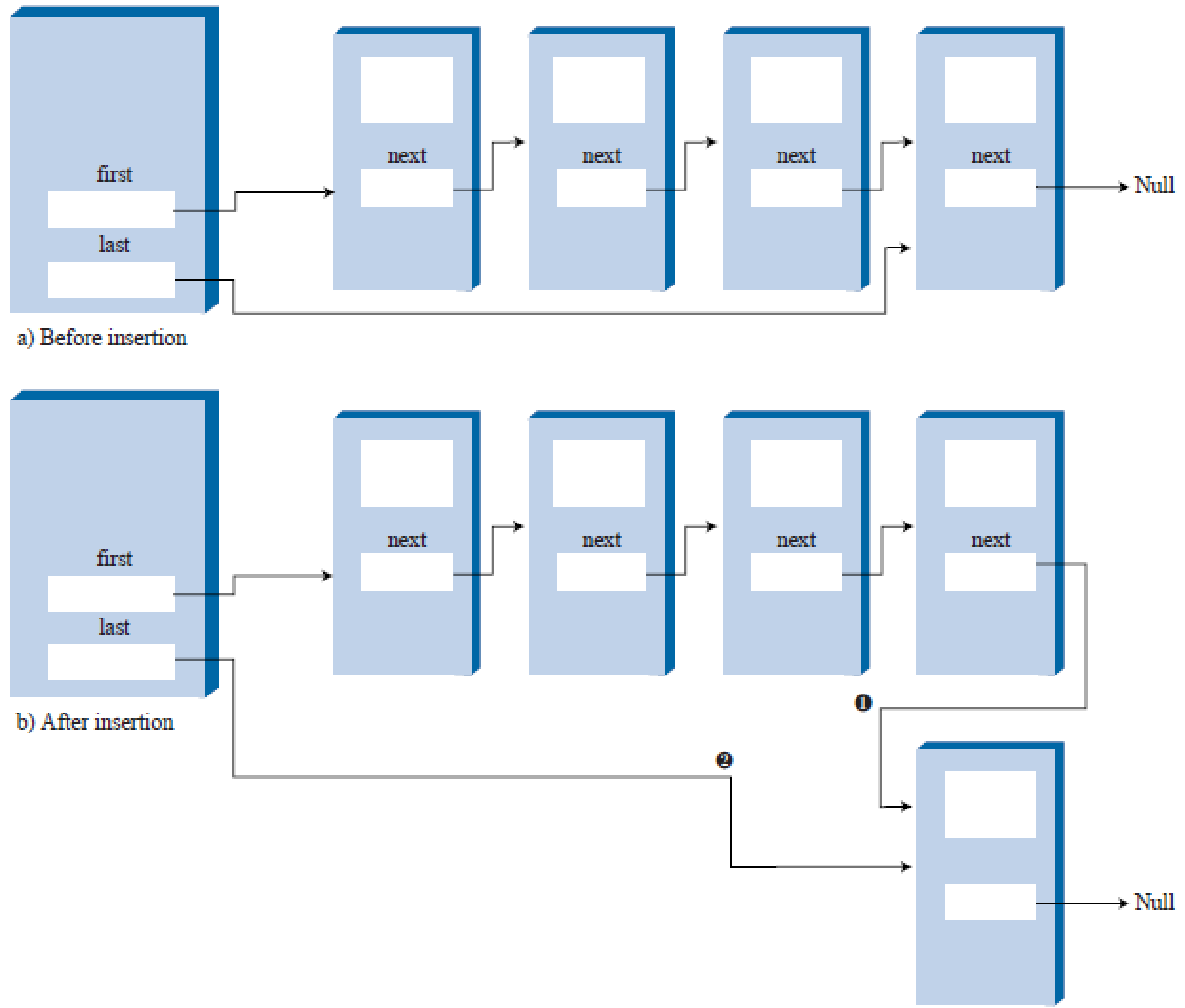
Double-Ended List

- It will be faster to implement Queue with Double-Ended Lists, during insertion element to Queue.
- Double-Ended Lists



Implement Queue Using Double-Ended List

- Operation Insertion

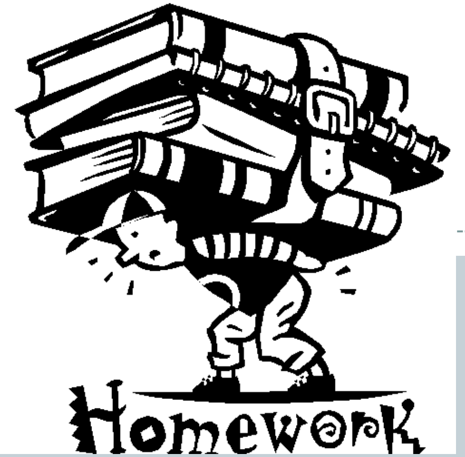


a) Before insertion

b) After insertion

```
class DELinkList{
    Link* pFirst;                  //ptr to first link on list
    Link* pLast;                   //ptr to the last link on list
public:
    void InitLinkList();
    bool IsEmpty();                //Return true – empty, else return false
    bool IsFull();                 //Return true – full, else return false
    void Remove ();//Remove the first link from list (in Q – Remove)
    void Insert (); //Insert the new link to the tail of list (in Q – Insert)
    Link *PeekFront(); //Return the first link on list
    int Size();    //Return the number link in the lists
};

void DELinkList::InitLinkList(){
    pFirst = NULL;
    pLast = NULL;                          //(no links on list yet)
};
```



Create Queue by Using Double-Ended Linked List with the following operations: Insert, Remove, PeekFront, Size, isEmpty, isFull

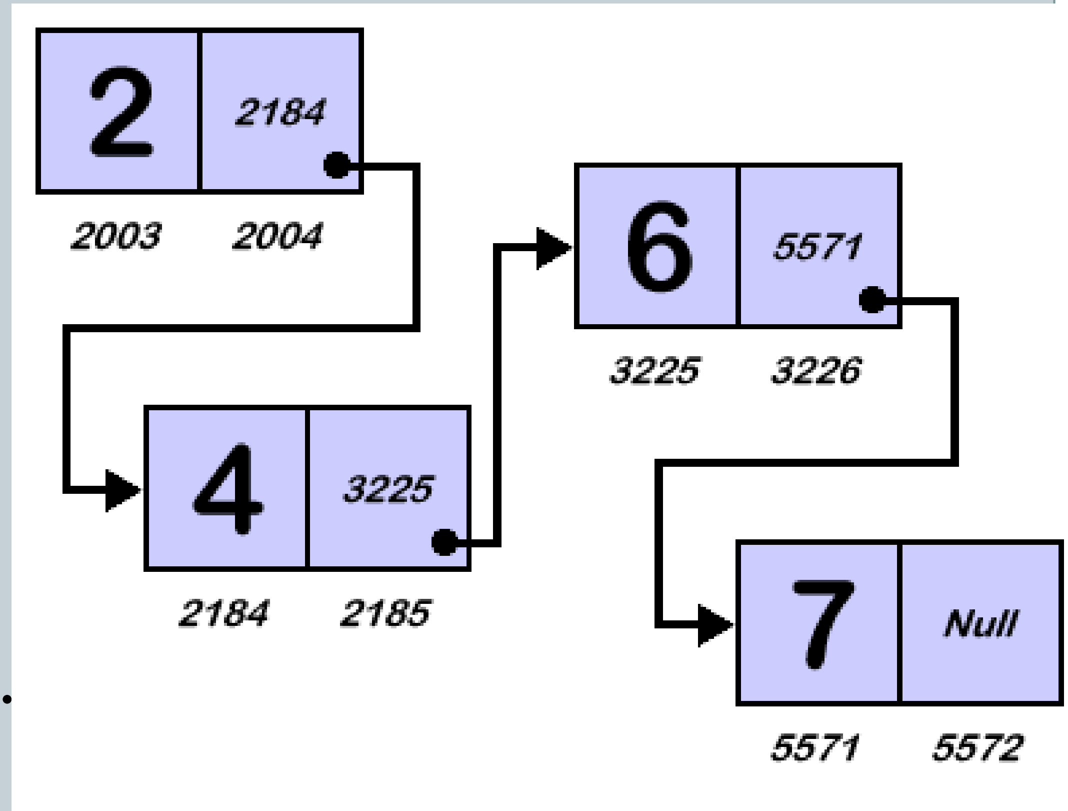
Outline

- Stacks
- Queues and Priority Queues
- Linked Lists
- Abstract Data Types
- Specialized Lists

Sorted Linked List

- In a sorted list, the items are arranged in sorted order by key value.

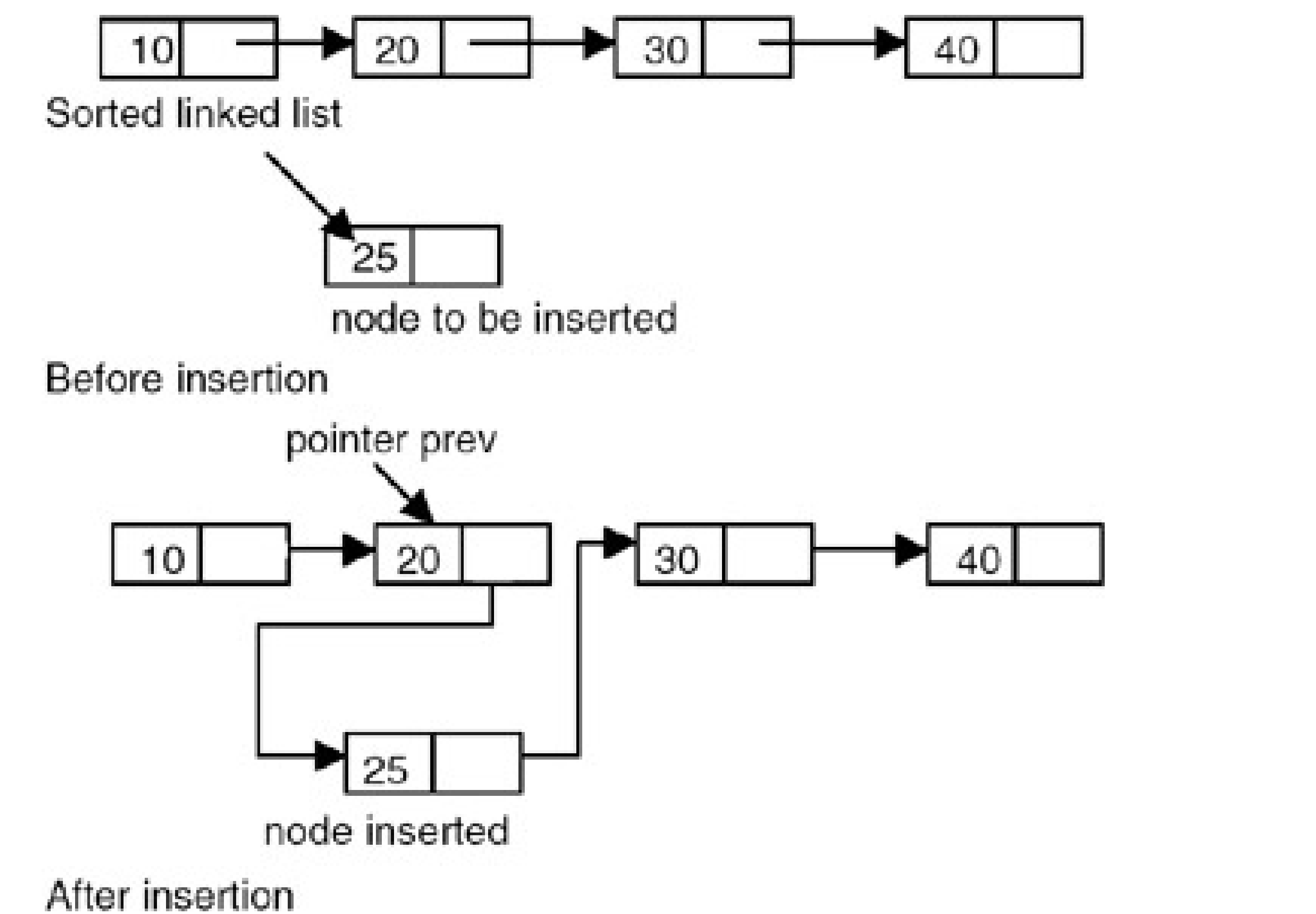
- The advantages of a sorted list over a sorted array are speed of insertion because elements don't need to be moved and the fact that a list can expand to fill available memory, whereas an array is limited.

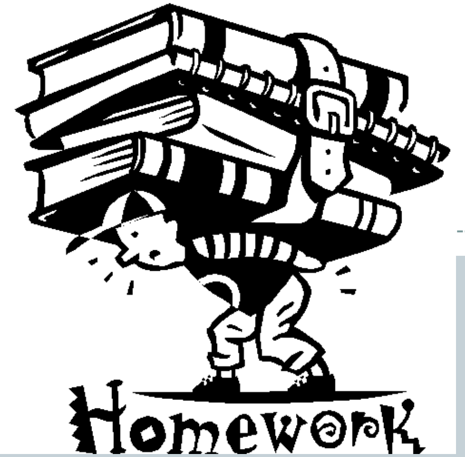


Implement the Sorted List

- Data of link in the list will be sorted, when we arrange them during the operation ***insertion***.
- The insertion function is more complicated for sorted lists than for unsorted lists.
- To insert an item into a sorted list, the algorithm must first search through the list until it finds the appropriate place to put the item.

Insertion Link to the Sorted Linked List





Create Sorted Linked List with the following operations: InsertInOrder, RemoveFirst, Size, isEmpty, isFull

To be continued…