15CSE374 INTRODUCTION TO DATA STRUCTURES AND ALGORITHMS

Sarath tv

Last Lecture

- Recursion.

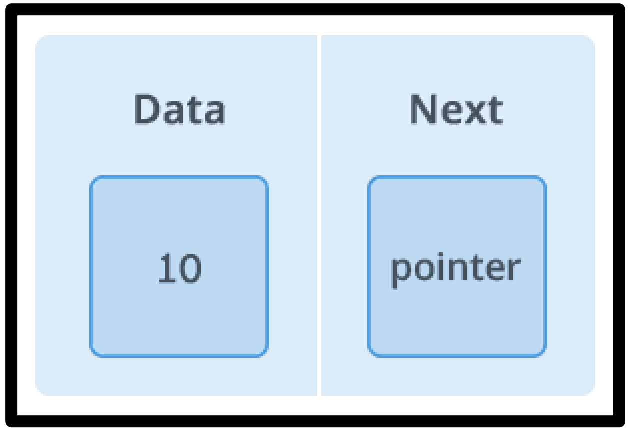
Linked List

- Common alternative to arrays in the implementation of data structures.
- Each item in a linked list contains a data element of some type and a pointer/reference to the next item in the list.
- It is <u>easy to insert and delete elements </u>in a linked list, which are not natural operations on arrays, since arrays have a fixed size.

- Linked lists and arrays are similar since they both store collections of data.
- Arrays are convenient to declare and the provide the handy [ ] syntax to access any element by its index number.
- **Disadvantages of arrays**
- The size of the array is fixed
- Most convenient thing for programmers to do is to allocate arrays which seem "large enough“
- Inserting new elements at the front is potentially expensive because existing elements need to be shifted over to make room.

- Linked lists have their own strengths and weaknesses, but they happen to be strong where arrays are weak.
- An array allocates memory for all its elements lumped together as one block of memory.
- In contrast, a linked list allocates space for each element separately in its own block of memory called a "linked list element" or "node".
- The list gets is overall structure by using pointers/reference to connect all its nodes together like the links in a chain.
- Each node contains two fields: a "**data**" field to store whatever element type the list holds for its client, and a "**next**" field which is a pointer used to link one node to the next node.
- The front of the list is a pointer to the first node

Node

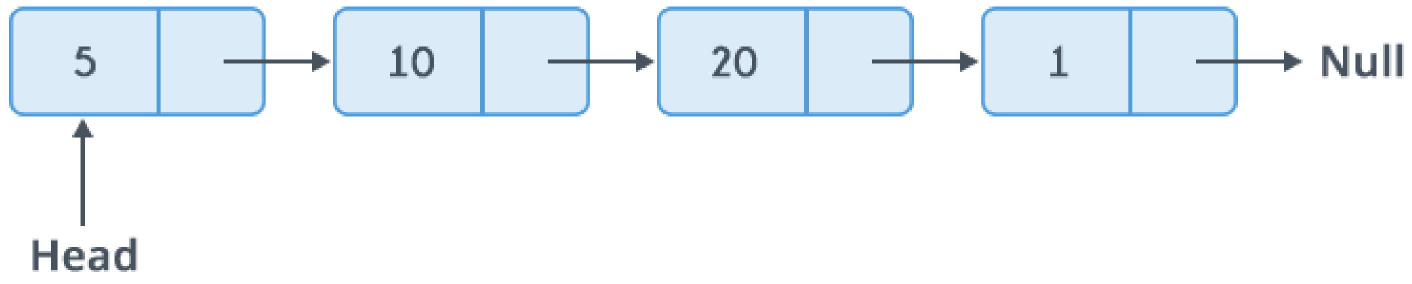


- A **linked list** is a way to store a collection of elements. Like an array these can be character or integers. Each element in a linked list is stored in the form of a **node**

A node is a collection of two sub-elements or parts. A **data** part that stores the element and a **next** part that stores the link to the next node.

Members in a linked list

- A linked list is formed when many such nodes are linked together to form a chain. Each node points to the next node present in the order. The first node is always used as a reference to traverse the list and is called **HEAD**. The last node points to **NULL**.



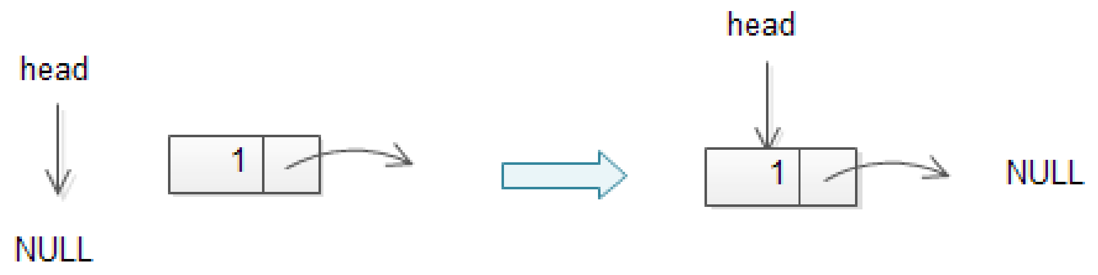
Creating a linked list

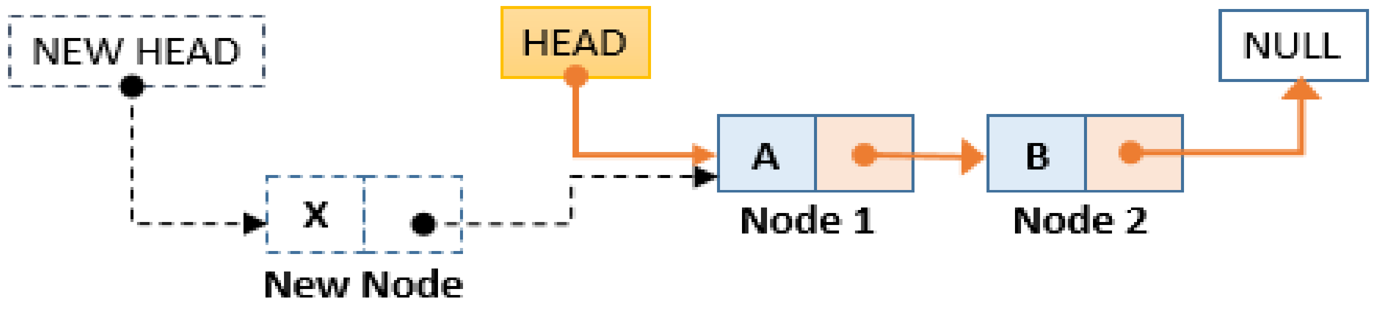
- Nodes
    - Each node will have two parts- data and pointer to next node.

Operation on linked List

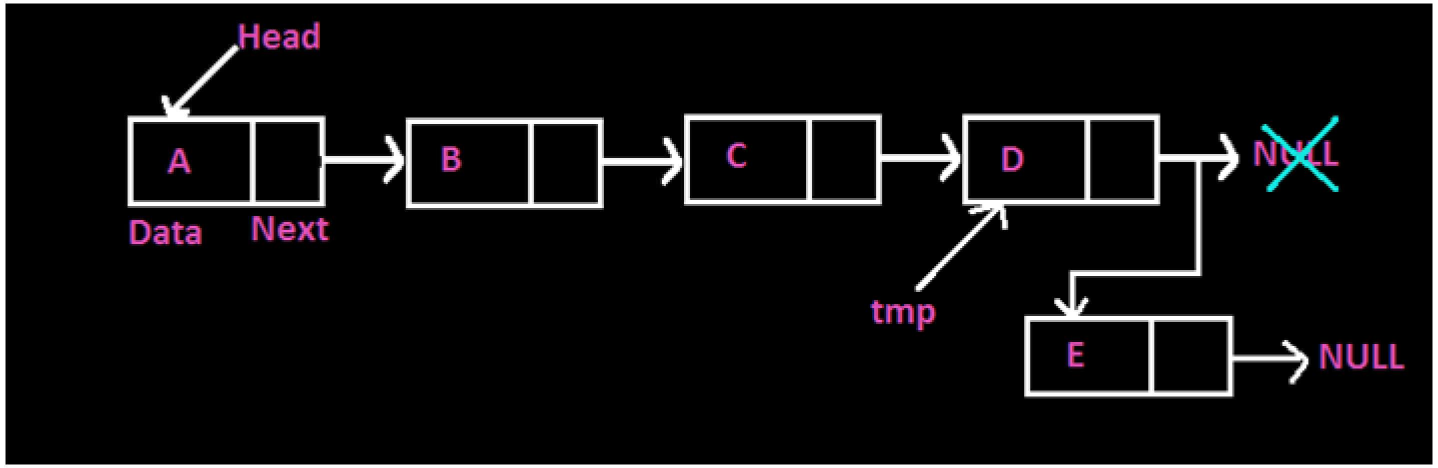
- Important points to remember:
- ➢Head points to the first node of the linked list
- ➢Next pointer of last node is NULL, so if next of current node is NULL, we have reached end of linked list.
- ➢All new node creation will be done from heap memory.
- ➢HEAD =NULL implies empty list.

Insert a data at the beginning of linked list

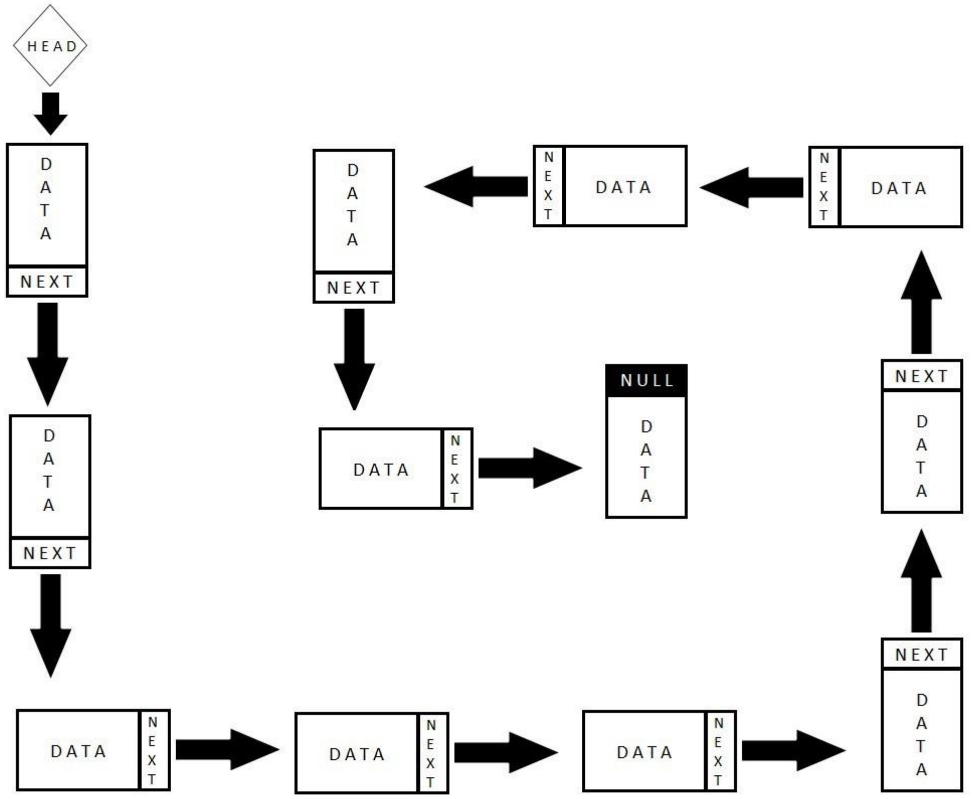




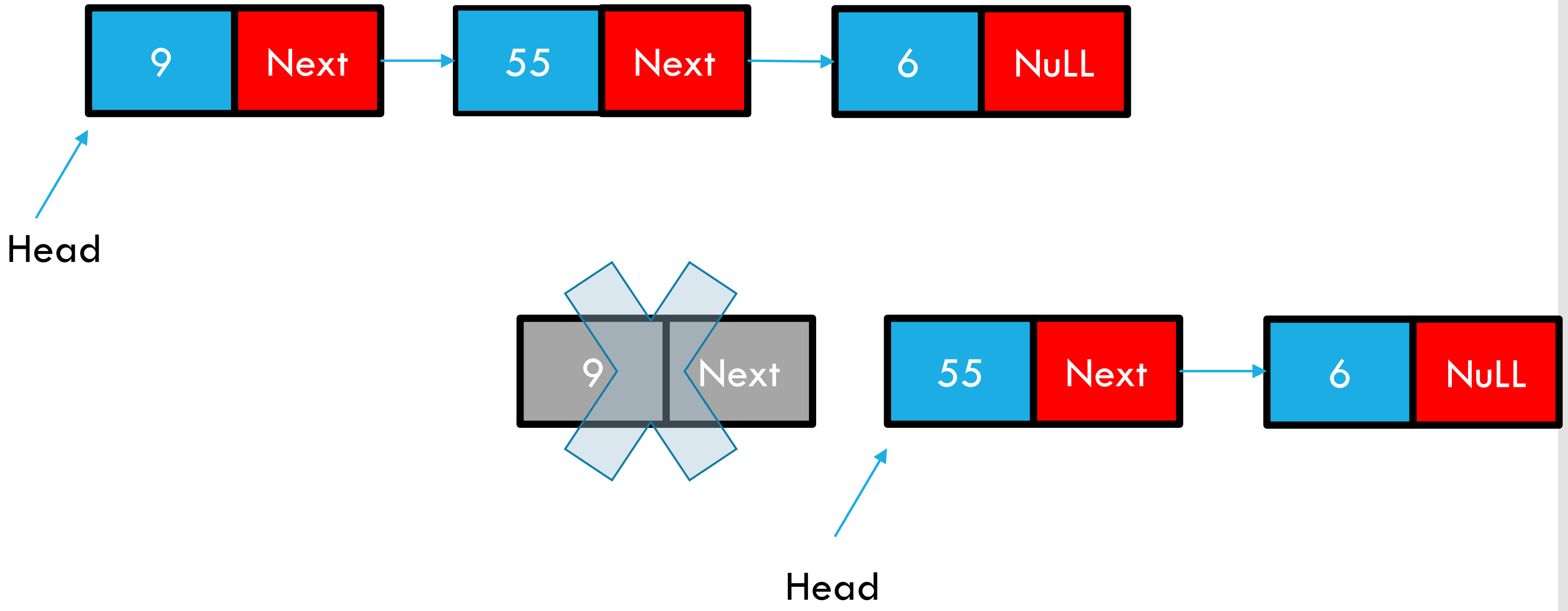
Insert a data at the End of linked list



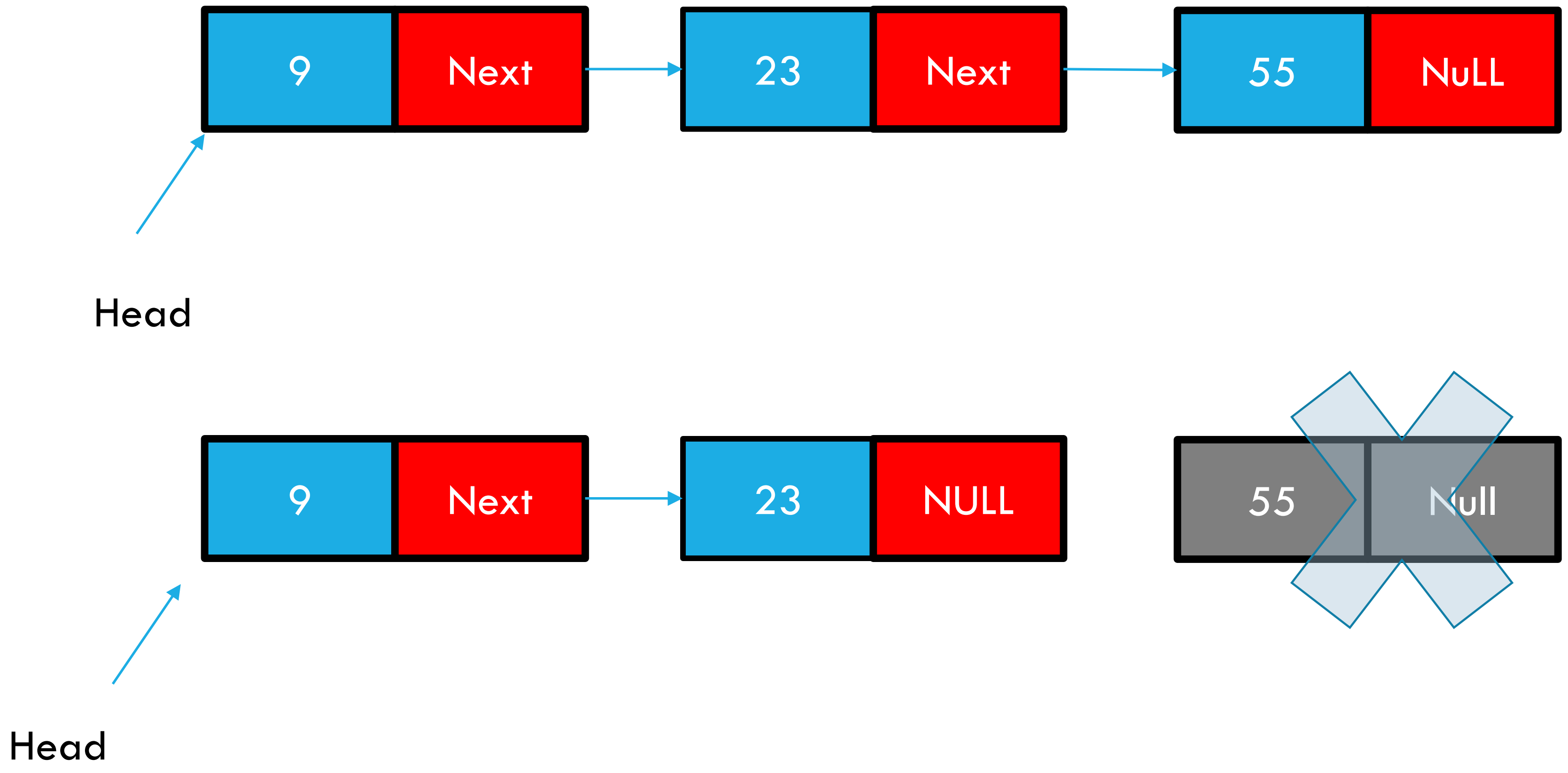
How to traverse a linked list



Deleting a node from beginning



Deleting a node from end



- No node
- Only one node
- Multiple nodes

THANK YOU!!!!!