

# 1.00 Tutorial 11

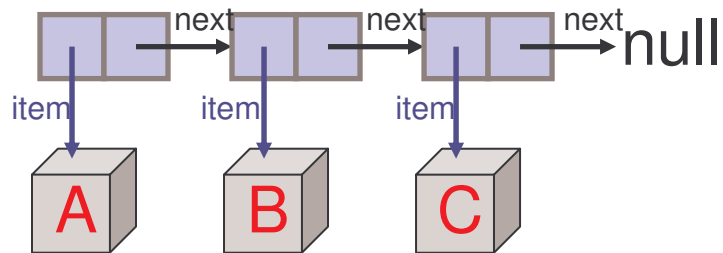
November 21-22, 2005

# Content

- 1D Linked List Recap
- Trees

# Linked-list: Recap

## Conceptual Diagram



## Linked-List (basically a list):

- contains a collection of element (object)
- arbitrary length
- cf. ArrayList in Java

## Basic operations

- addFirst/addLast
- removeFirst/removeLast
- traverse/print

## List Interface (from lecture)

```
public interface List {  
    public boolean isEmpty();  
    public void addFirst( Object o );  
    public void addLast( Object o );  
    public boolean contains(Object o);  
    public Object removeLast()  
        throws NoSuchElementException;  
    public Object removeFirst()  
        throws NoSuchElementException;  
    public boolean remove(Object o);  
    public void clear();  
    public int size();  
    public void print();  
}
```

# Linked Lists

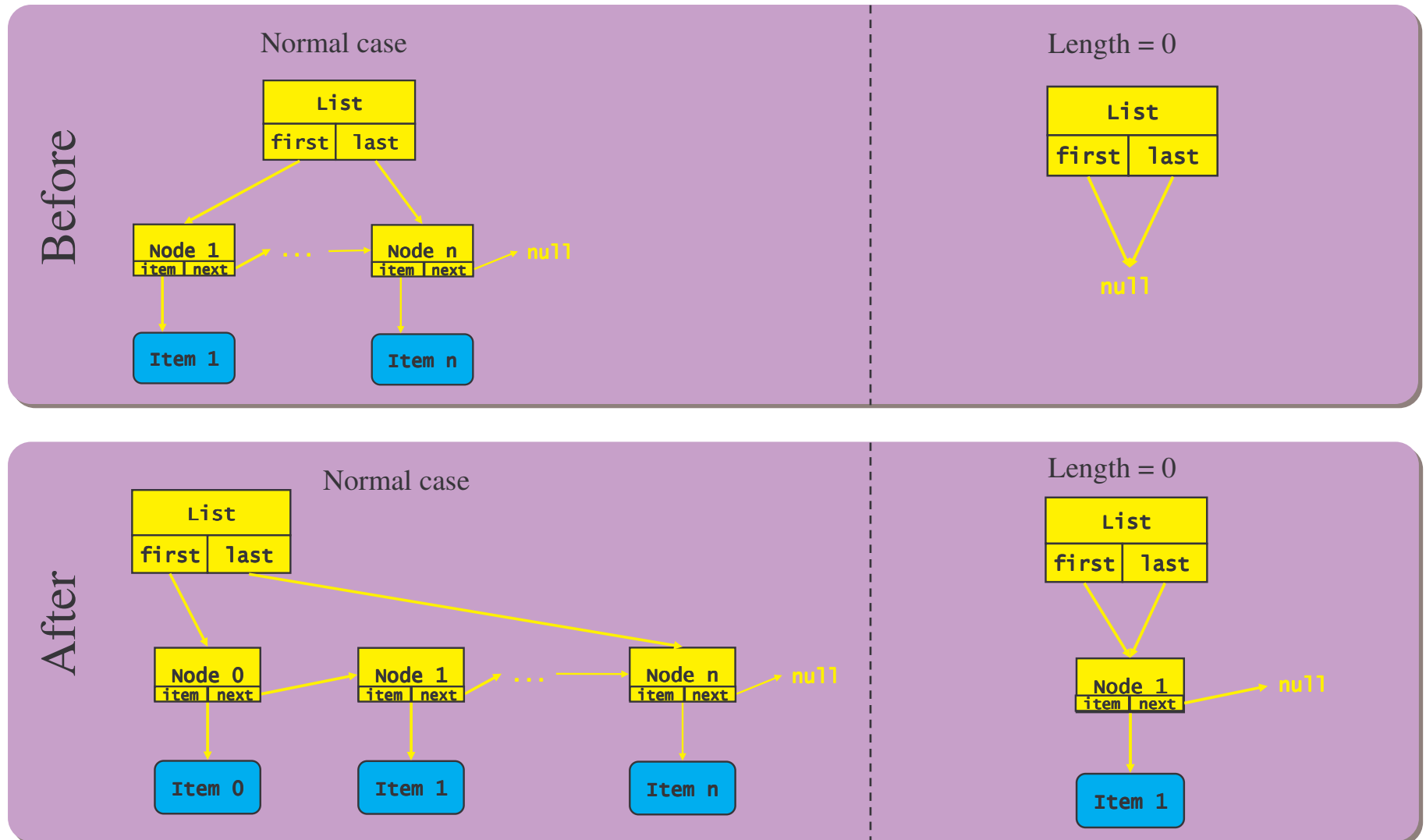
- A `LinkedList` is made of a series of “nodes”, each with:
  - an associated data object
  - a “pointer” (reference) to the next node of the list
- Traverse the list by following each node’s next pointer

# Linked-list: Tips

- Always think of special cases
  - What if your link is empty?
  - What if there is only one element?
- Always draw a diagram
- Warming up!
  - What are special cases for addFirst method?/Draw a diagram
  - What are special cases for removeLast method?/Draw a diagram

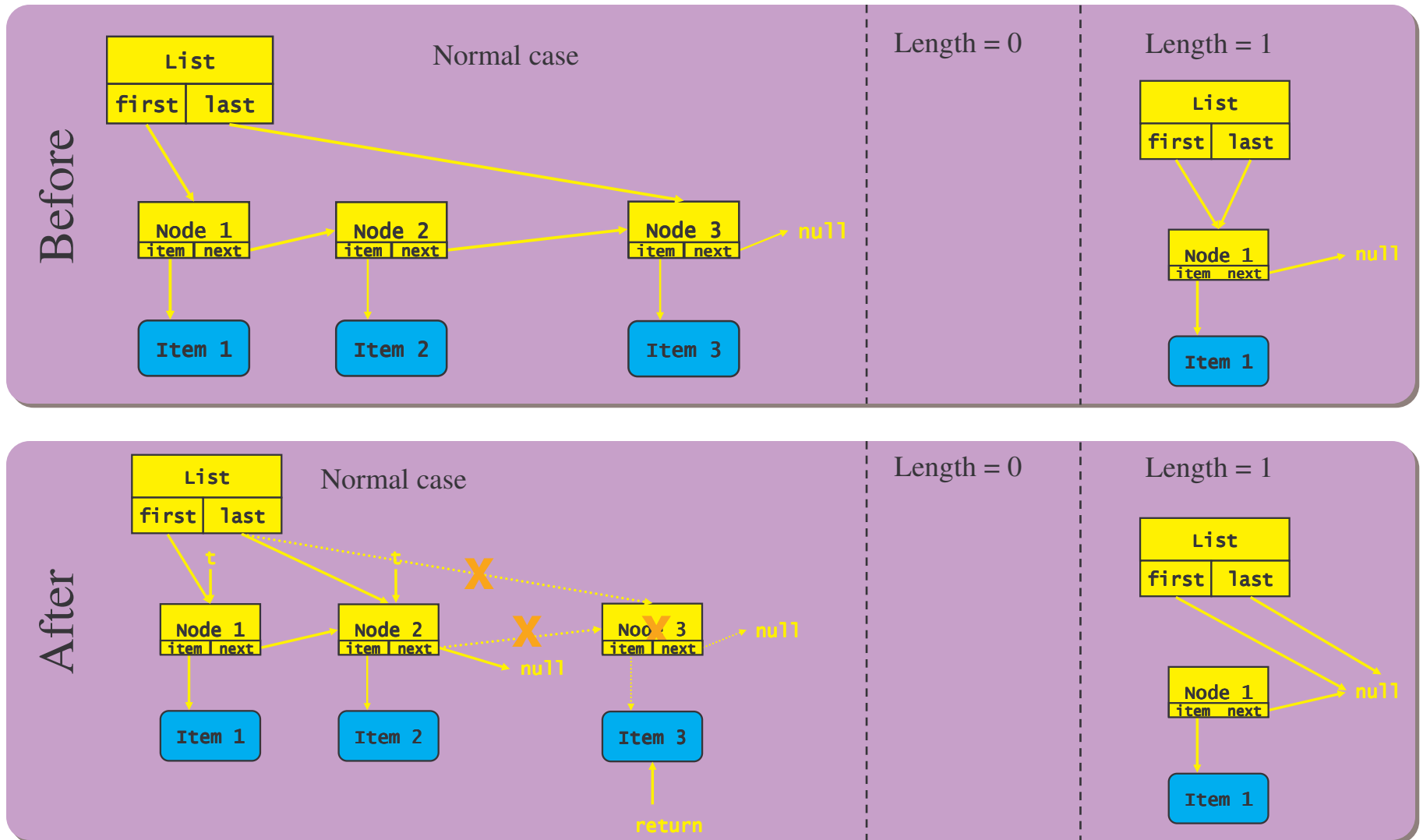
# Linked-list: Answers to Warm-up Questions

## addFirst Method



# Linked-list: Answers to Warm-up Questions

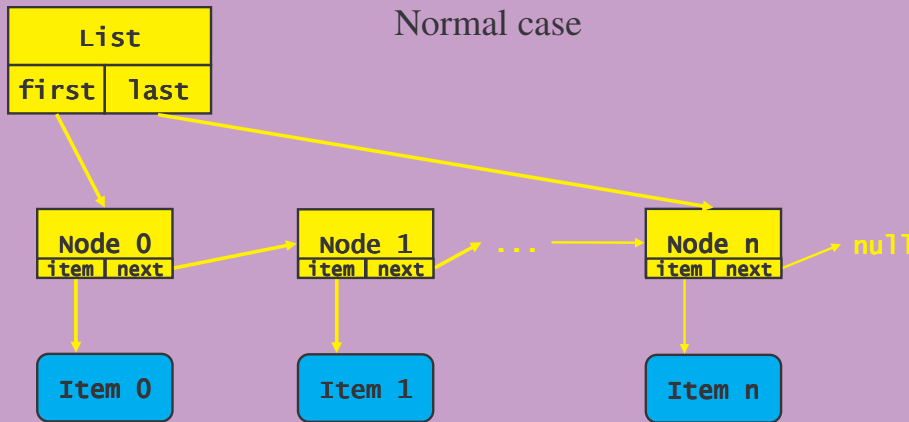
## removeLast Method



# Linked-list: insertAt(int i)

## insertAt Method

Before

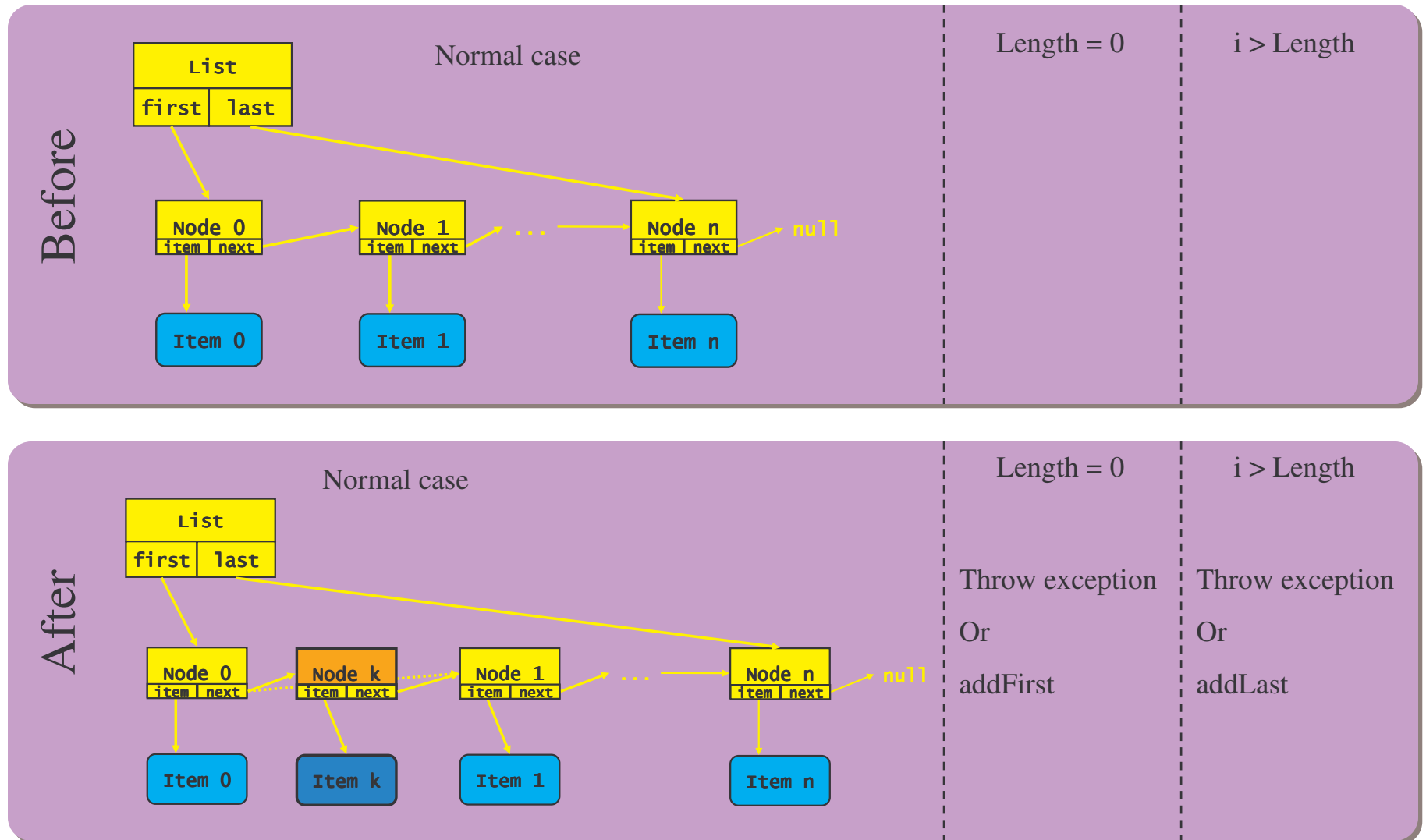


After

Normal case

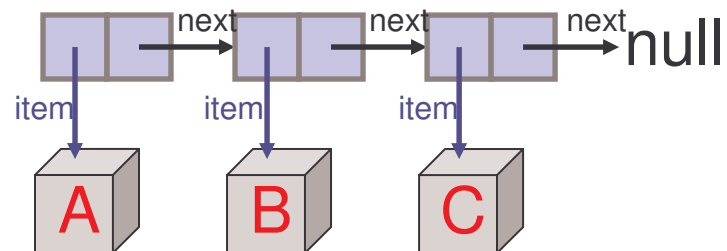
# Linked-list: insertAt(int i)

## insertAt Method



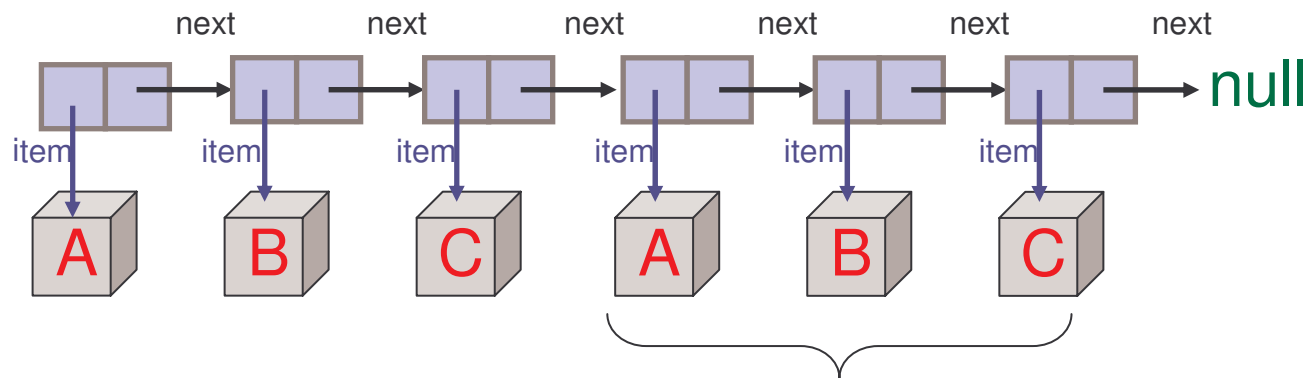
# Linked List Problem

- Using the linked list classes in Tutorial 11 on the Web site, write the method `repeatElementsAtEnd()` in the `Tutorial11.java` file that will add each element in the list to the end.
- *i.e.*, take this list



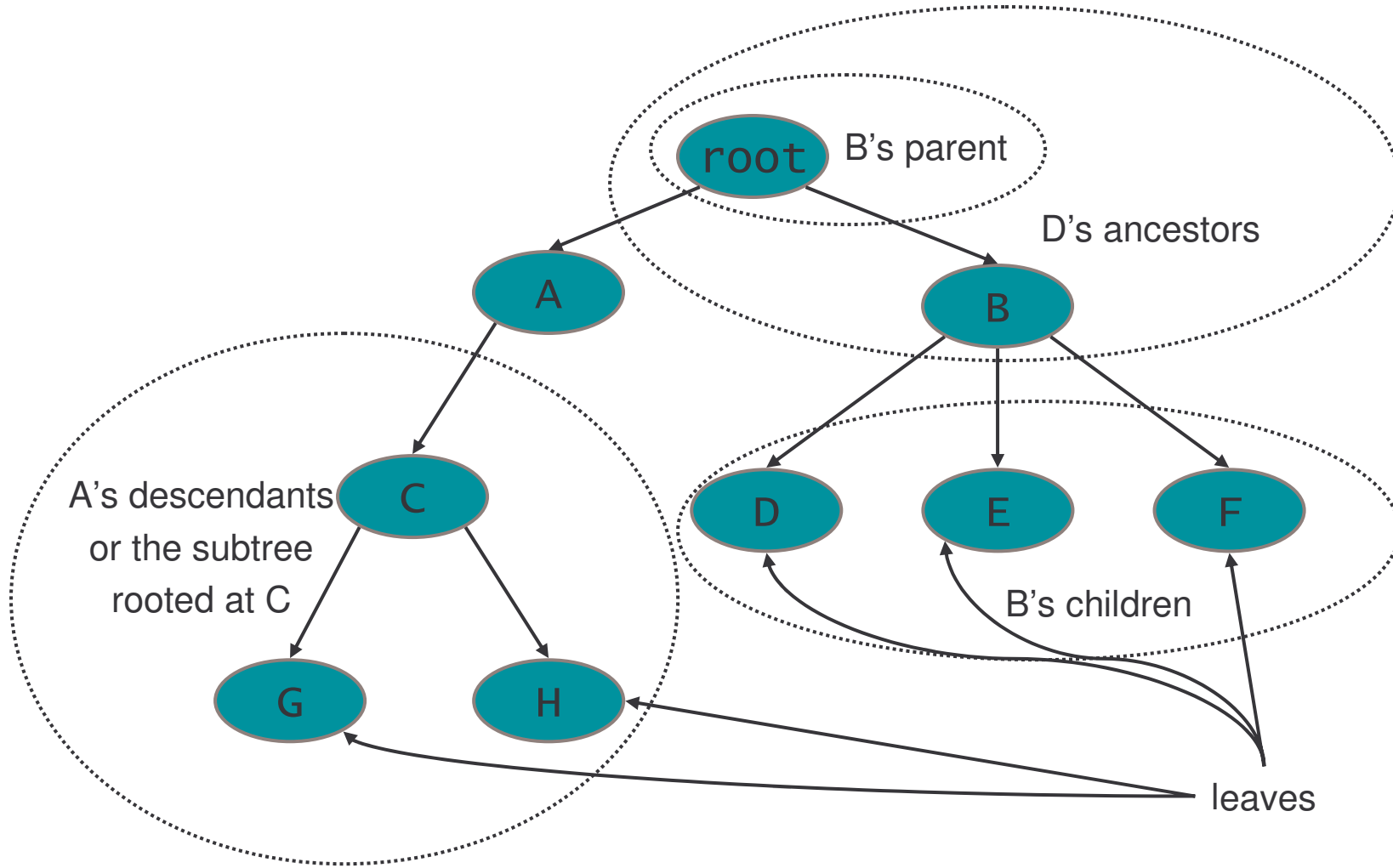
# Linked List Problem

- And make it look like this: (by reproducing each element in the list in order and adding it to the end.)

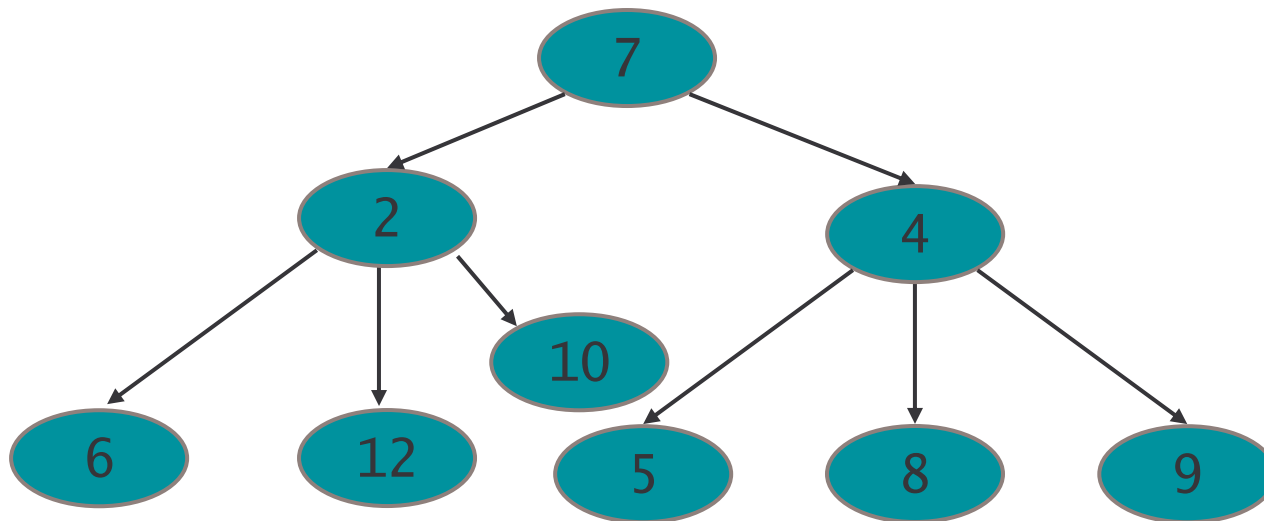


Three new nodes have been added

# Tree Terminology



# Question



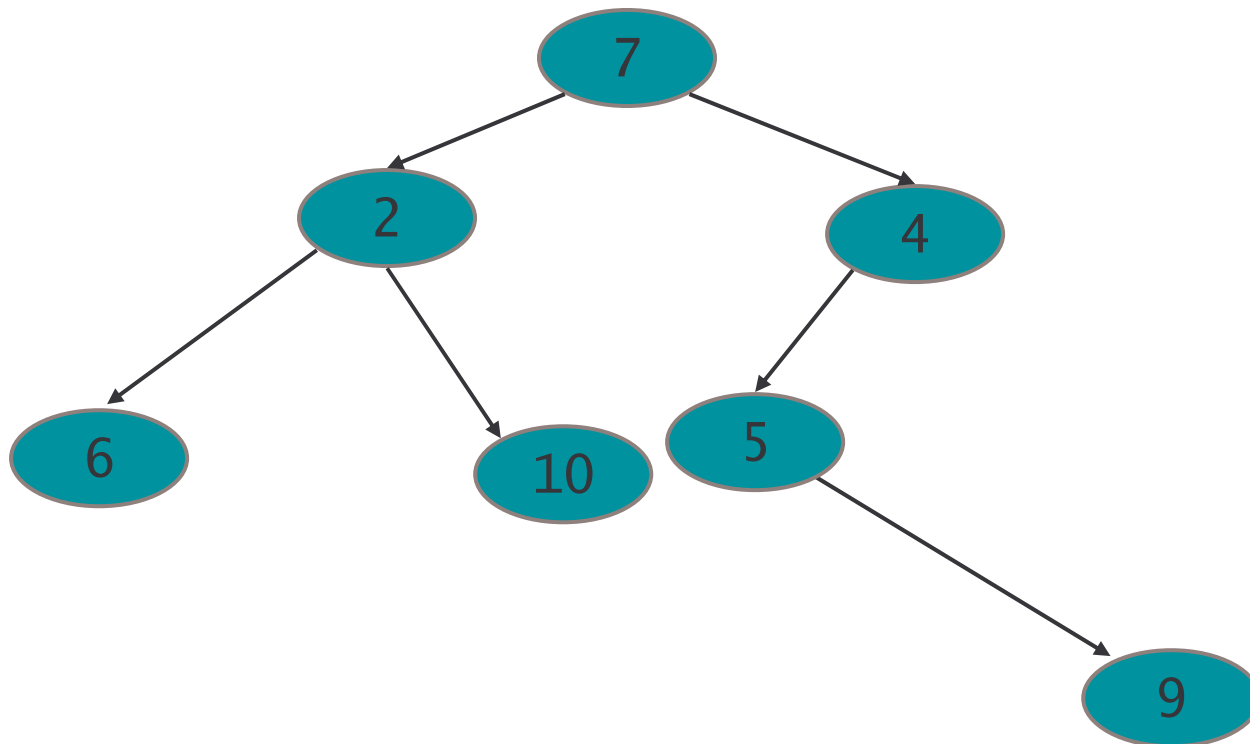
Which is the root node?

List the ancestors of the node 5

Which are the leaf nodes?

# Binary tree

- Binary tree is a tree with each node having at most 2 children



# Tree Traversal

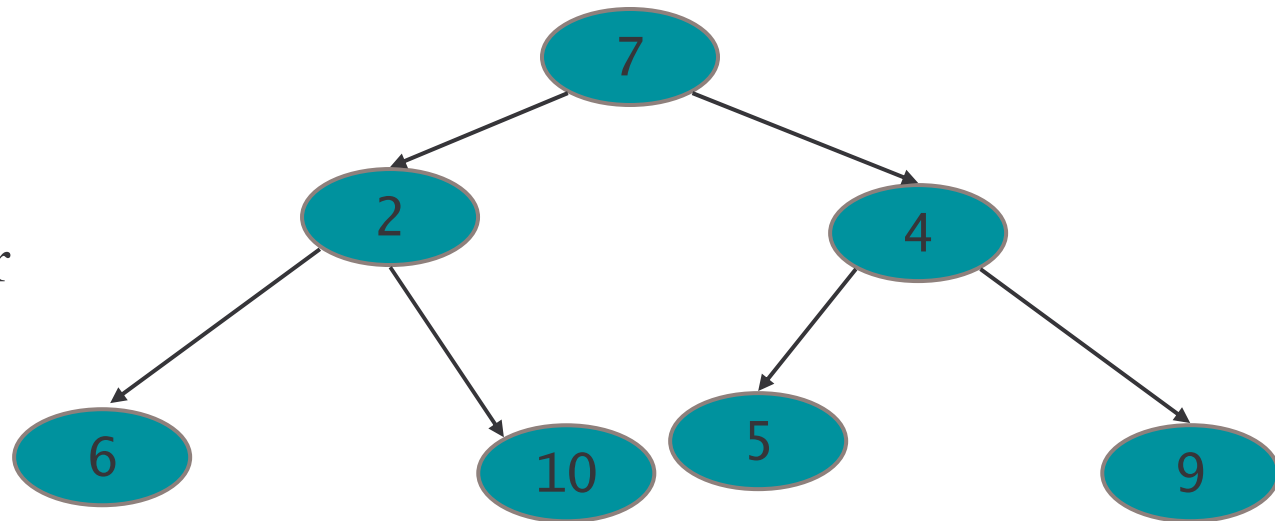
Visiting all nodes of a binary tree just once

- Pre-Order: Root, Left, Right
- In-Order: Left, Root, Right
- Post-Order: Left, Right, Root

# Traversal example

List the nodes when they are traversed in

- a) Pre-Order
- b) In-Order
- c) Post-Order



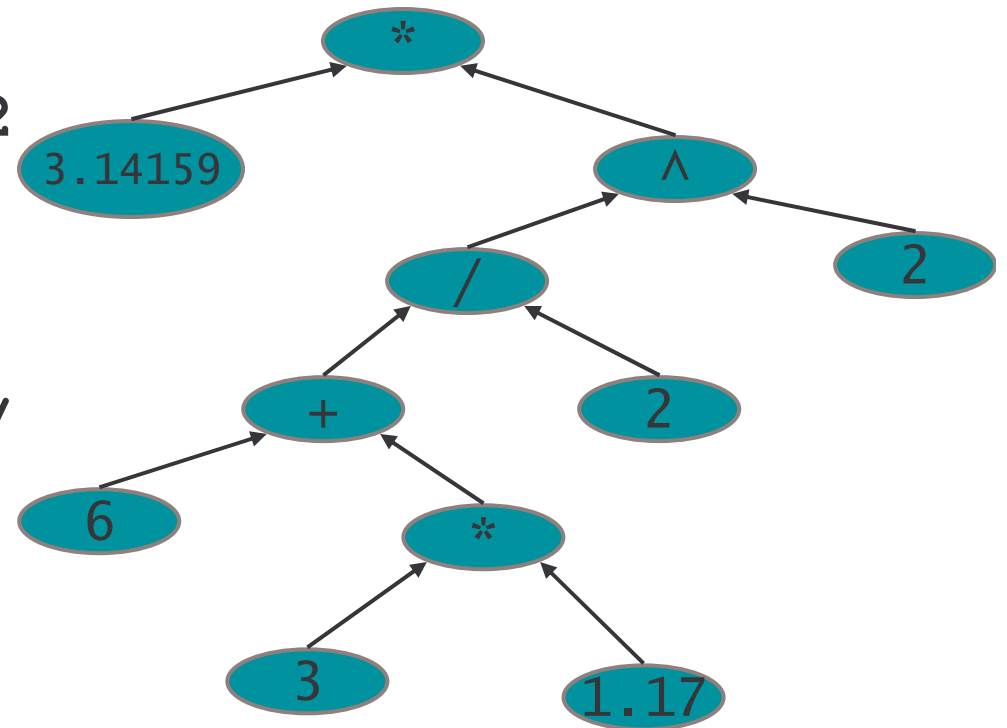
# Parse Tree Example

Original expression

$3.14159 * ((6 + 3 * 1.17) / 2) ^ 2$

Post order traversal of  
the parse tree

3.14159 6 3 1.17 \* + 2 /  
2 ^ \*



# Keys and Values

- A key is ordered i.e A key can be compared with another object of the same type
- A node in an ordered binary tree consists of an ordered key and a value
- Keys of different nodes should be of the same type

# Binary Search Trees

- The left subtree of every node (if it exists) must only contain nodes with keys less than or equal to the parent and the right subtree (if it exists) must only contain nodes with keys greater than or equal to the parent.

- In-Order traversal visits the nodes in ascending order

- **Exercise:**

Create a BST of 5, 3, 7, 9, 2, 4, 8, 1. Verify with a In-Order Traversal. What is the algorithm used in creating the BST?

Now create a BST of 1, 2, 3, 5, 7, 8, 9. What happened?

ps89 questions?