LINKED LISTS

Chapter 1

OBJECTIVES

In this lecture we are going to cover:

- 1.7 A linked list class

  Operations: create, getHead, isEmpty, equals, copy & reverse

- 1.8 Palindrome

  How to use linked lists to check for palindromes

- 1.9 Merging lists

  How to merge two sorted linked lists

1.7 A LINKED LIST CLASS

You will find uploaded with this lecture the Java code of the class (data structure) LinkedList including all the methods (operations). Make sure you download it and run it.

In this section we will look at the methods that are not explained in the other sections. These operations are: create (constructor), getHead, isEmpty, equals, copy , and reverse.

1.7 A LINKED LIST CLASS

```
public class LinkedList {
   private Node head; // a reference to the beginning of the list

    // This constructor creates an empty linked list
    public LinkedList() {
        head = null;
    }
   // This method returns a reference (link) to the head node of the list
    public Node getHead() {
        return head;
    }
    //This method checks if a list is empty or not
    public boolean isEmpty()     {
        return head == null;
    }
```

1.7 A LINKED LIST CLASS

```
// This method checks if two LinkedList are equal
public boolean equals(LinkedList secondList) {
    if (countNodes() != secondList.countNodes()) // Different number of nodes
        return false;
    // If the two lists are of equal length, start from the head of each list
    Node curr1 = head, curr2= secondList.head;
    while (curr1 != null) {  // or curr2 != null
      // node in list1 has different data than corresponding node in list 2?
        if (curr1.data.compareTo(curr2.data)!=0)
            return false;
        curr1 = curr1.next; // go to the next node in list 1
        curr2 = curr2.next; // go to the next node in list 2
      } // end while
     return true;
    } // end equals
```

1.7 A LINKED LIST CLASS

```
// This method copies the content of list 1 to list 2
public LinkedList copy() {
    LinkedList list2 = new LinkedList(); // list 2 is empty
    Node curr1 = head;
    while (curr1 != null)  {
        list2.addTail(curr1. data); //add at the end of list 2
        curr1 = curr1.next;
    } // end while
    return list2;
  } // end copy
```

1.7 A LINKED LIST CLASS

```
// This method reverses the elements of list 1 in list 2
public LinkedList reverse() { // This is the method explained in class
    LinkedList list2 = new LinkedList(); // list 2 is empty
    if (isEmpty()) // if list 1 is empty
        return list2;  // list 2 is also empty
    Node curr1 = head;
    while (curr1 != null)  {
          list2.addHead(curr1.data); //add at the end of list 2
          curr1 = curr1.next;
    } // end while
    return list2;
 } // end reverse
```

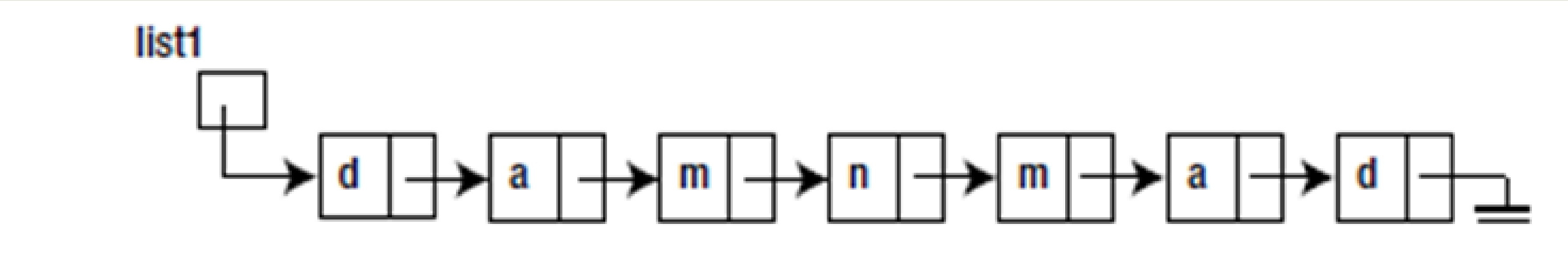
1.8 PALINDROME

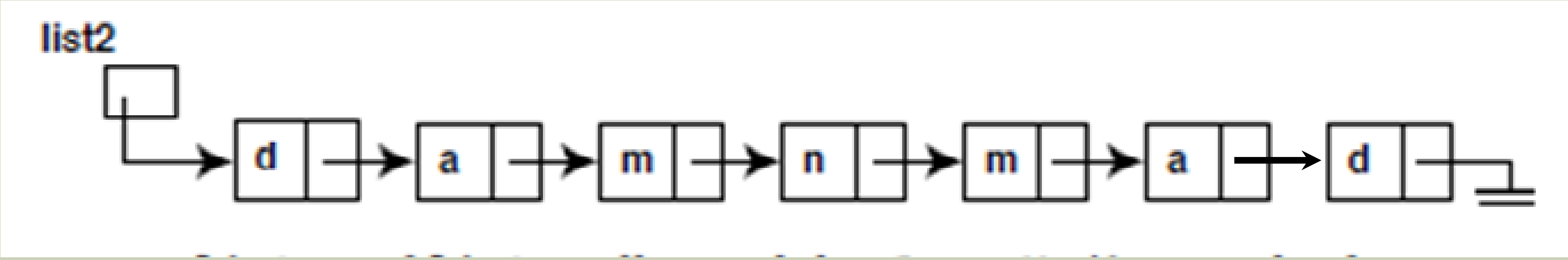
We want to learn how we can use linked lists to solve the problem of detecting palindromes. A palindrome is a word, number, or sequence that reads the same backwards as forwards, e.g. damnmad, madam, abcba, 1234321, or nursesrun.

To do so we follow the below steps:

1. Store the original phrase in a linked list called (list1), one character per node.
2. Reverse list1 and name the reversed list (list2).
3. Compare list1 with list2 using the method equals. If they are equal then we have a palindrome.

EXAMPLE: PALINDROME





PALINDROME CODE

```
public class Application {
  public static void main(String[]args) {
   Scanner scan = new Scanner(System.in);
   String s;
   LinkedList list1 = new  LinkedList();
   System.out.print("Enter a string: ");
   s = scan.next();
   //Change the type of data in NodeData from int to char
   for(int i=0;i<s.length();i++)
      list1.addTail(new NodeData(s.charAt(i)));
   if(isPalindrome(list1))
     System.out.println(s + " is a palindrome");
   else
     System.out.println(s + " is not a palindrome");
  }
  public static boolean isPalindrome(LinkedList list1) {
    LinkedList list2 = list1.reverse();
     return (list1.equals(list2));
  }
} // end Application
```

PALINDROME CODE

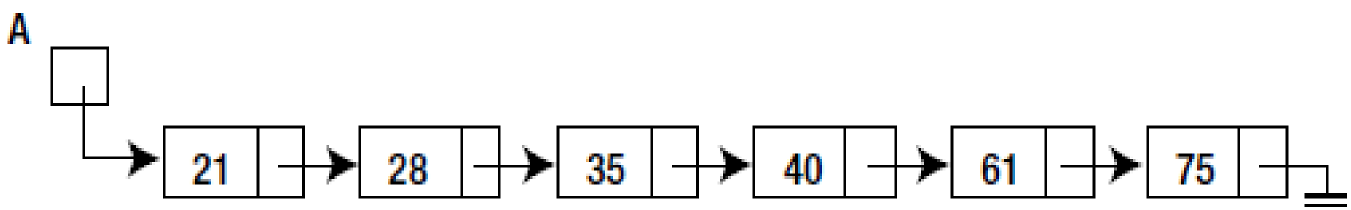
A complete java application that contains the solution for palindrome is also uploaded to the Google Classroom. Make sure you download it and run it.

1.9 MERGING LISTS

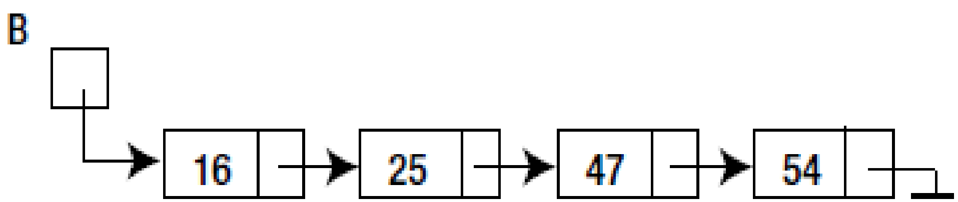
- How to merge two sorted linked lists

MERGING TWO SORTED LINKED LISTS

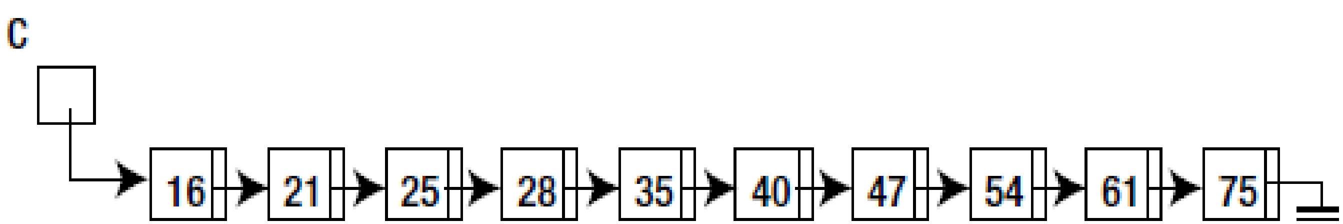
Suppose the given lists are as follows:



and



We want to create one linked list with all the numbers in ascending order, thus:



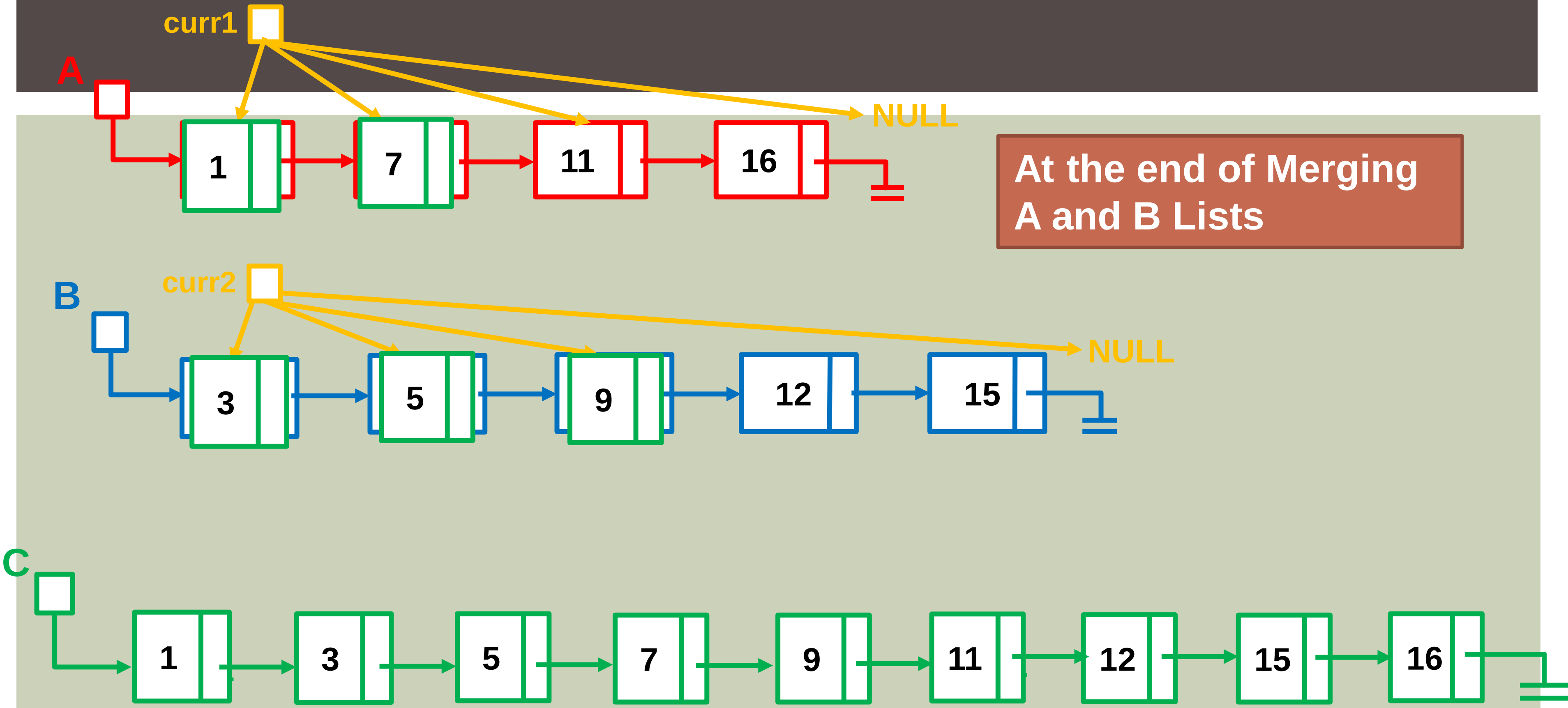
PSEUDOCODE

```
while (at least one number remains in both A and B) {
   if (smallest in A < smallest in B)
      add smallest in A to C
      move on to next number in A
   else
      add smallest in B to C
      move on to next number in B
   endif
}
//at this stage, at least one of the lists has ended
while (A has numbers) {
   add smallest in A to C
   move on to next number in A
}
while (B has numbers) {
   add smallest in B to C
   move on to next number in B
}
```

CODE

```
// This method merges two sorted lists. The new list is sorted too.
   public LinkedList mergeSorted(LinkedList secondList) {
      LinkedList mergedList = new LinkedList();
      Node curr1 = head, curr2= secondList.head;
      while (curr1 != null && curr2 != null) {
         if(curr1.data.compareTo(curr2.data) < 0) {
            mergedList.addTail(curr1.data);
            curr1 = curr1.next;
         }
         else {
            mergedList.addTail(curr2.data);
            curr2 = curr2.next;
         }
      } // end while
      // In case the first list is longer (it still has nodes)
      while (curr1 != null) {
          mergedList.addTail(curr1.data);
          curr1 = curr1.next;
      }
      // In case the second list is longer (it still has nodes)
      while (curr2 != null) {
          mergedList.addTail(curr2.data);
          curr2 = curr2.next;
      }
      return mergedList;
   } // end mergeSorted
```

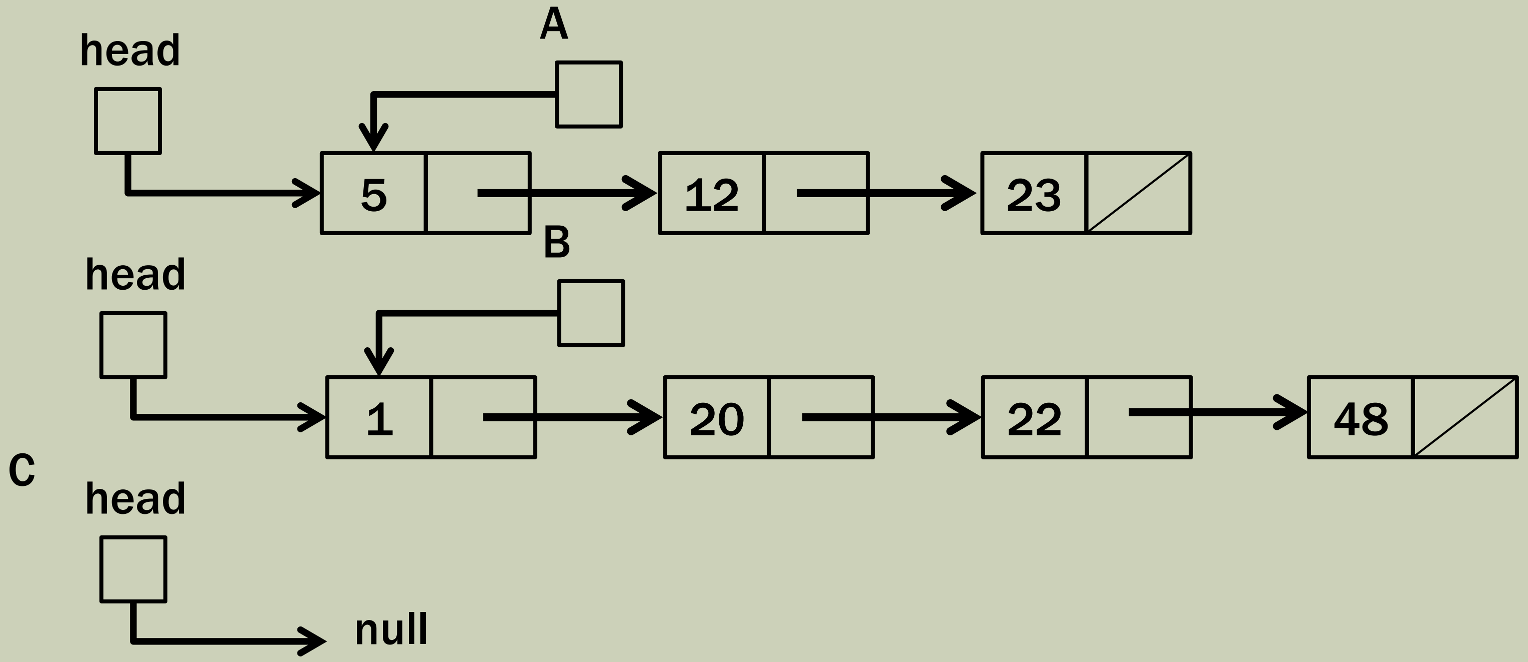
Example 1: DEMO WITH ANIMATION



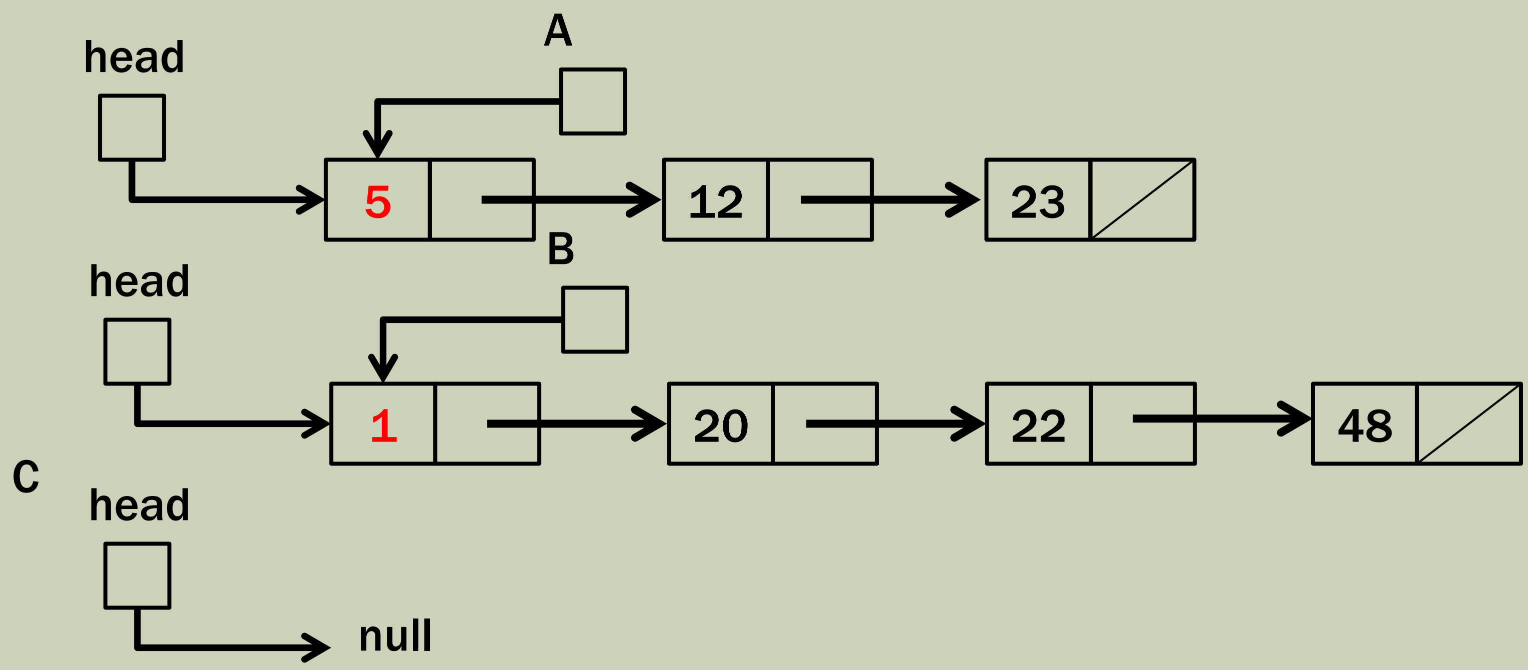
At the end of Merging A and B Lists

Click (and keep clicking) to see the animation.

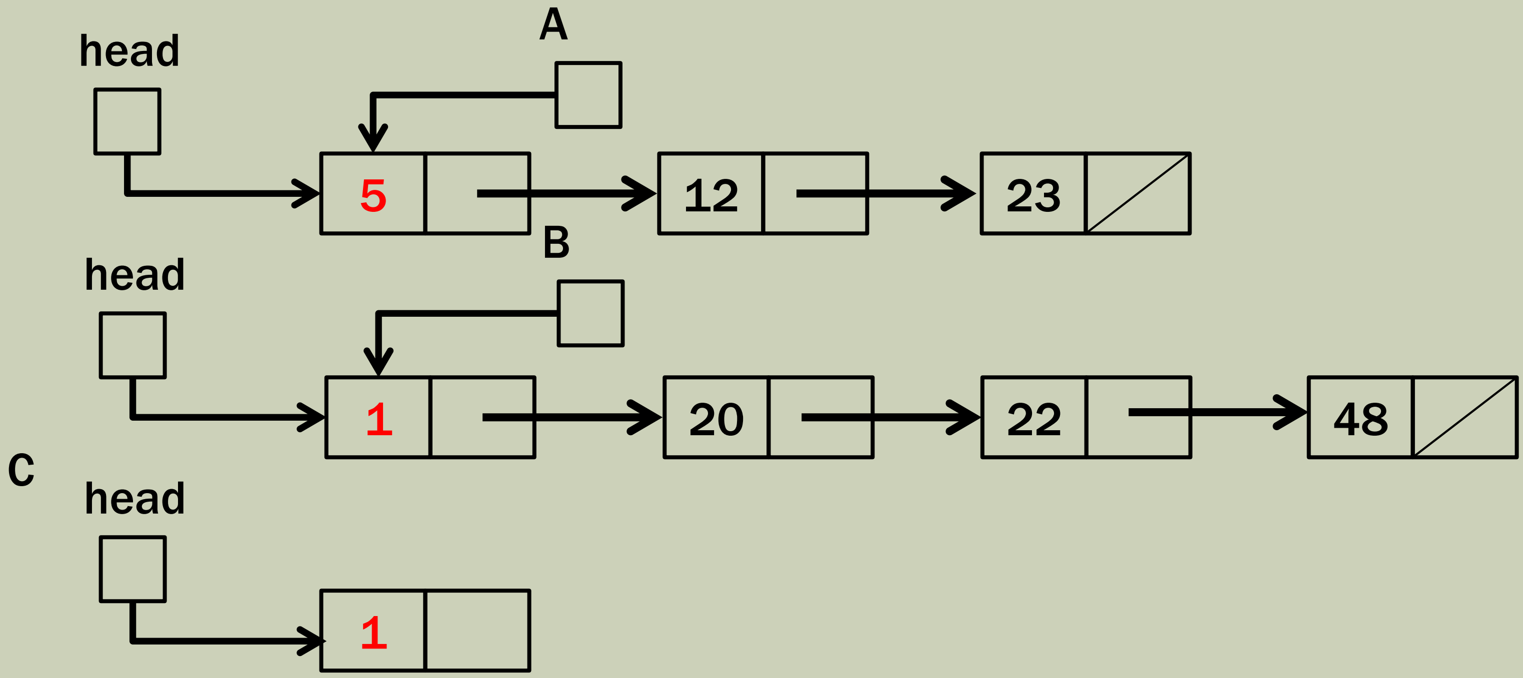
EXAMPLE 2



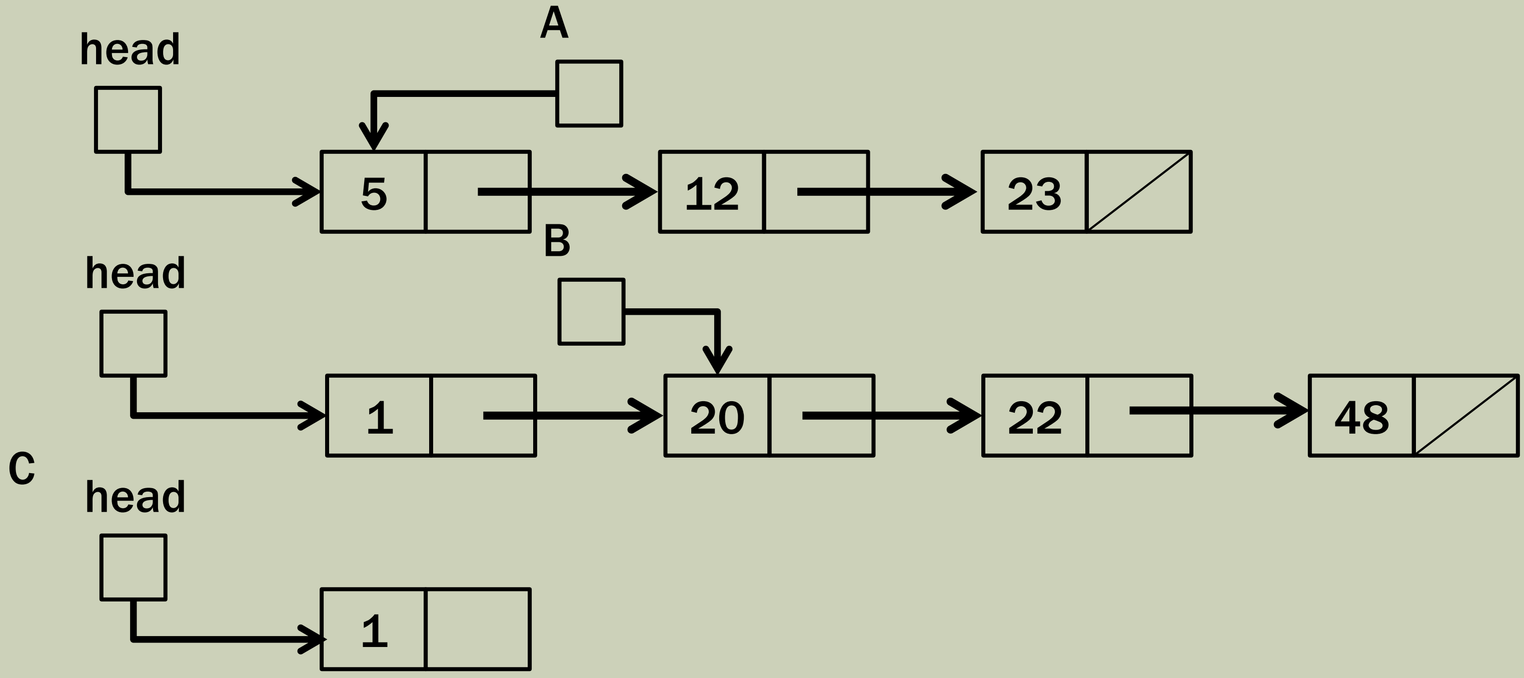
EXAMPLE 2 (CONTINUED)



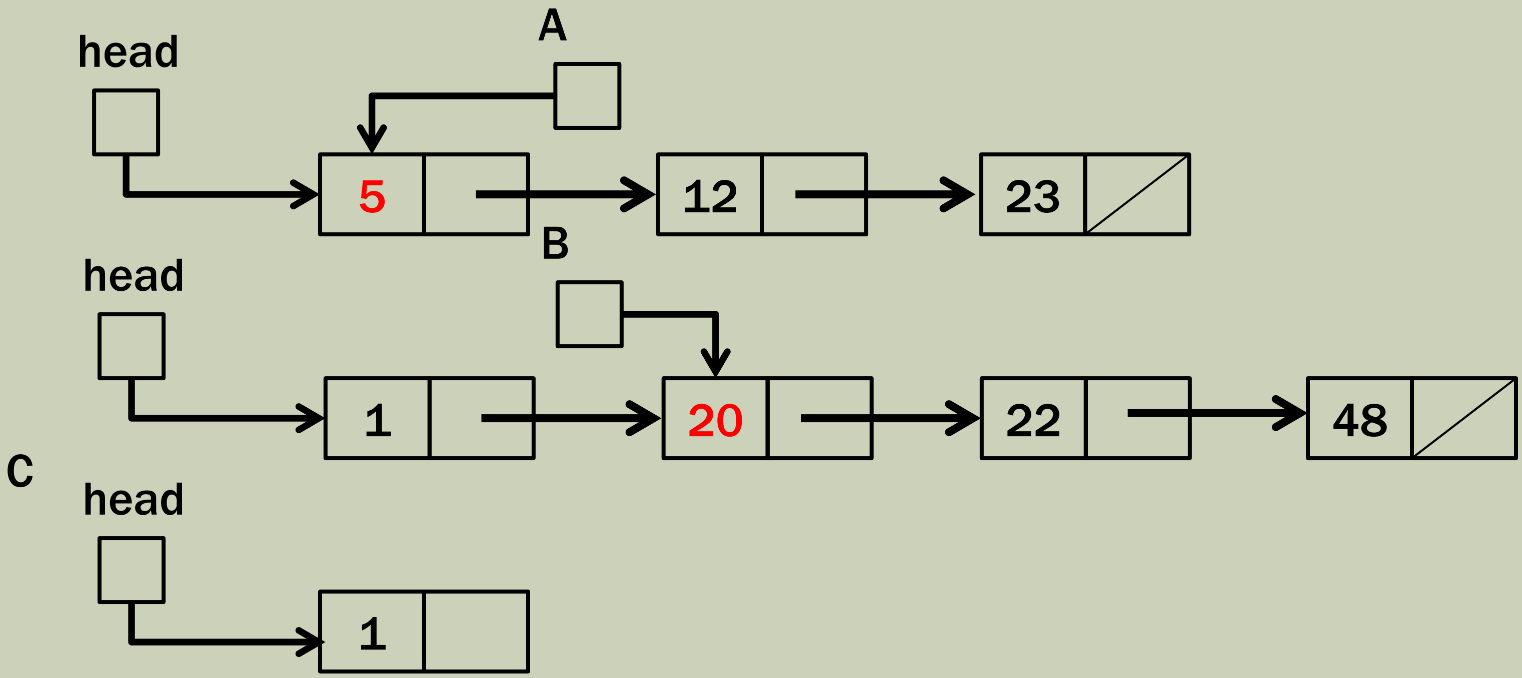
EXAMPLE 2 (CONTINUED)



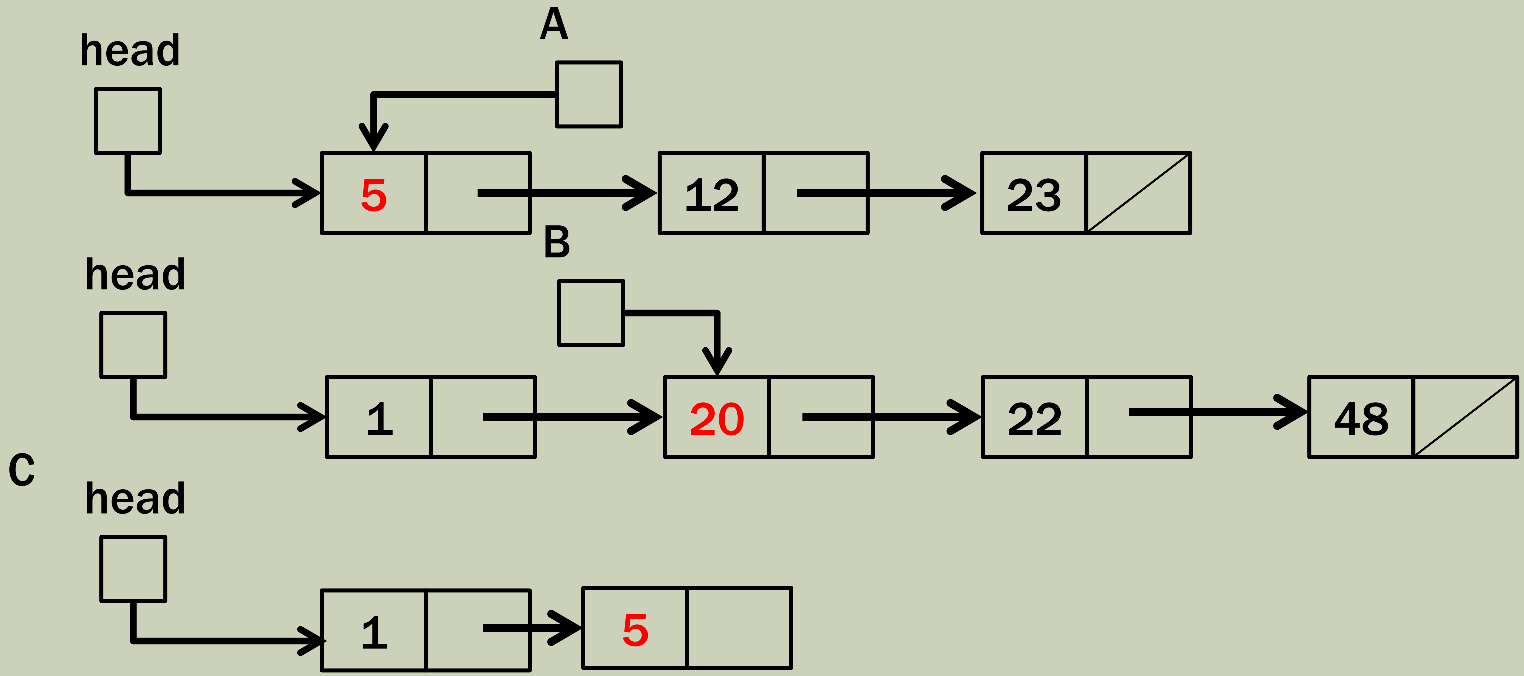
EXAMPLE 2 (CONTINUED)



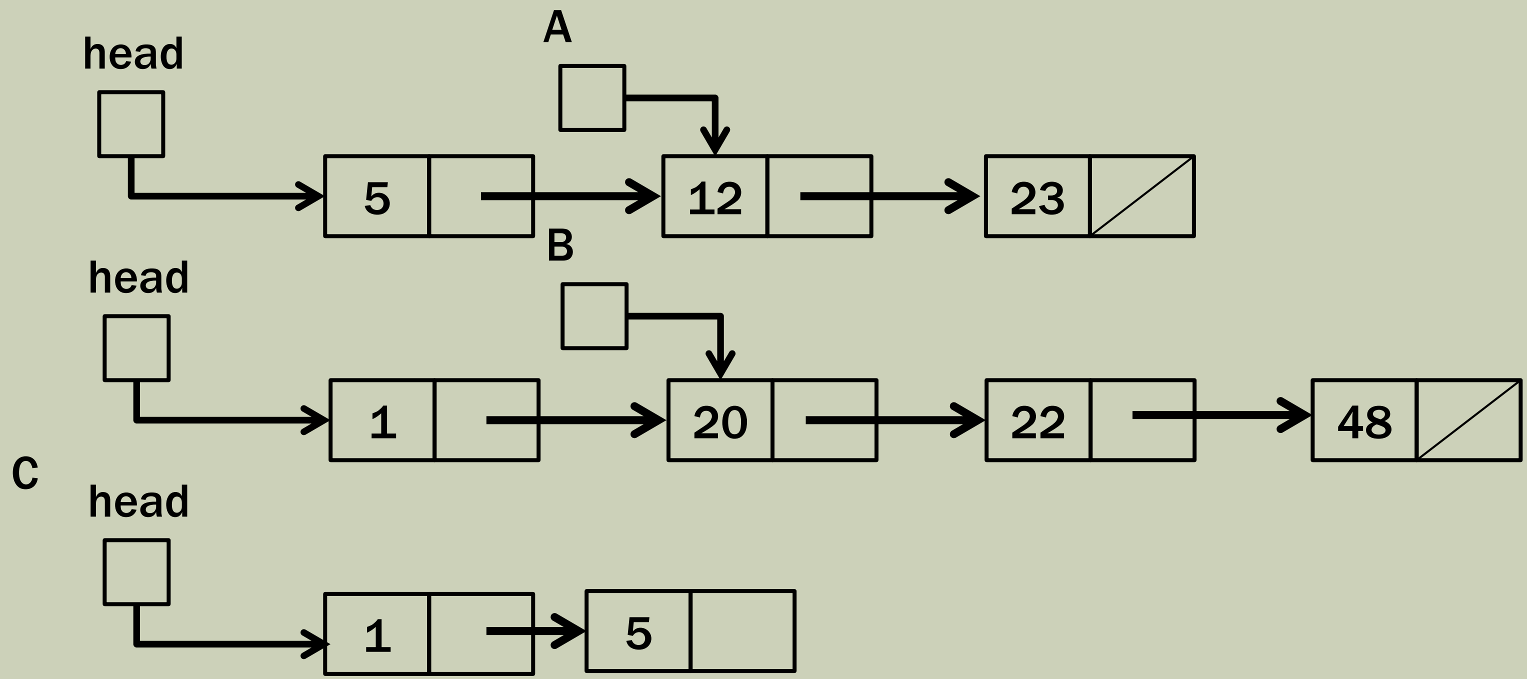
EXAMPLE 2 (CONTINUED)



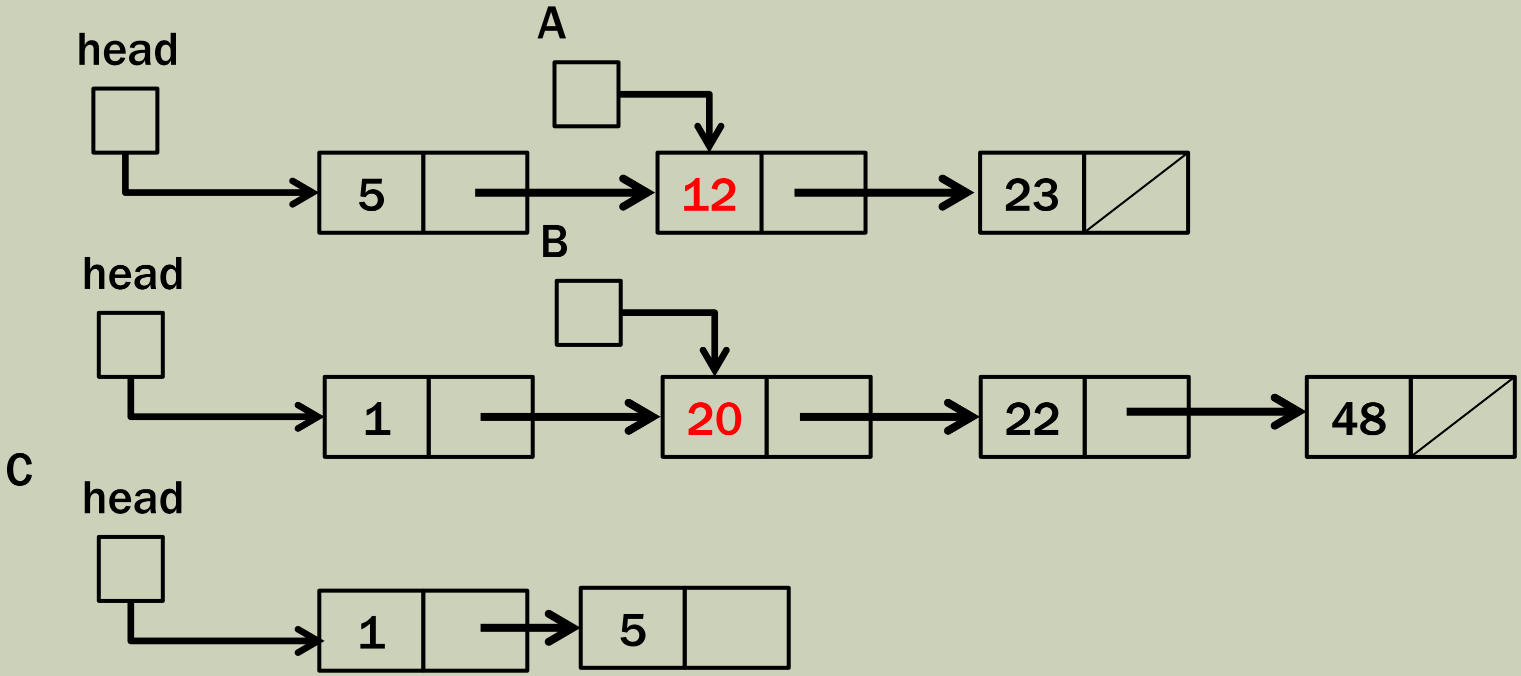
EXAMPLE 2 (CONTINUED)



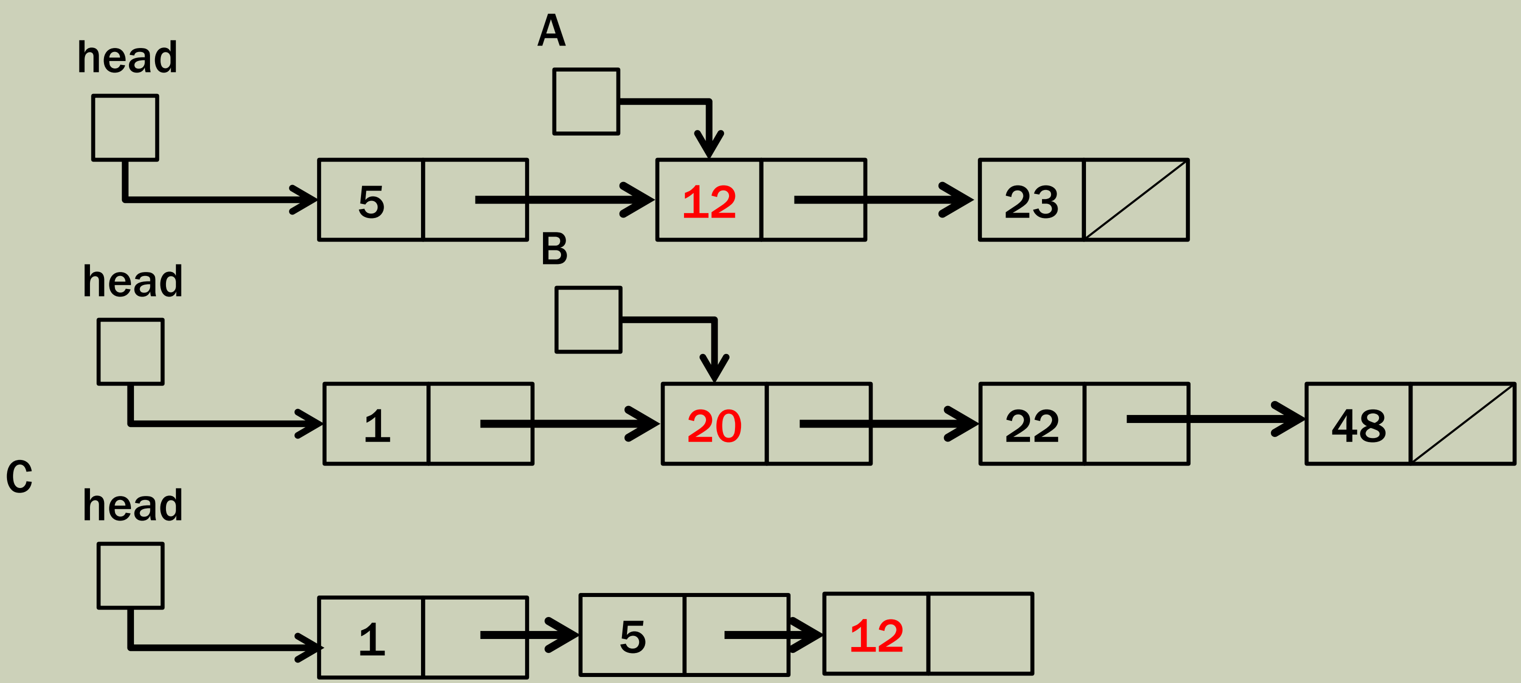
EXAMPLE 2 (CONTINUED)



EXAMPLE 2 (CONTINUED)



EXAMPLE 2 (CONTINUED)



EXAMPLE 2 (CONTINUED)

And so on…