CMPT 280

Topic 5: Doubly Linked Node Chains and Lists

Mark G. Eramian

University of Saskatchewan

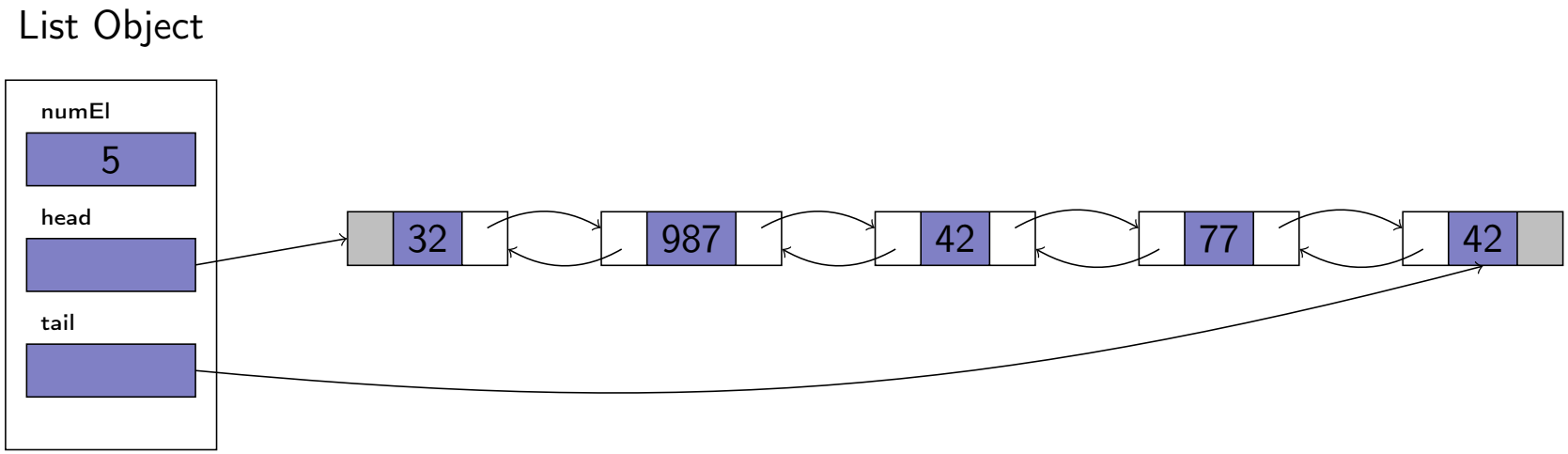
References

- Textbook, Chapter 5

Doubly Linked Node Chains

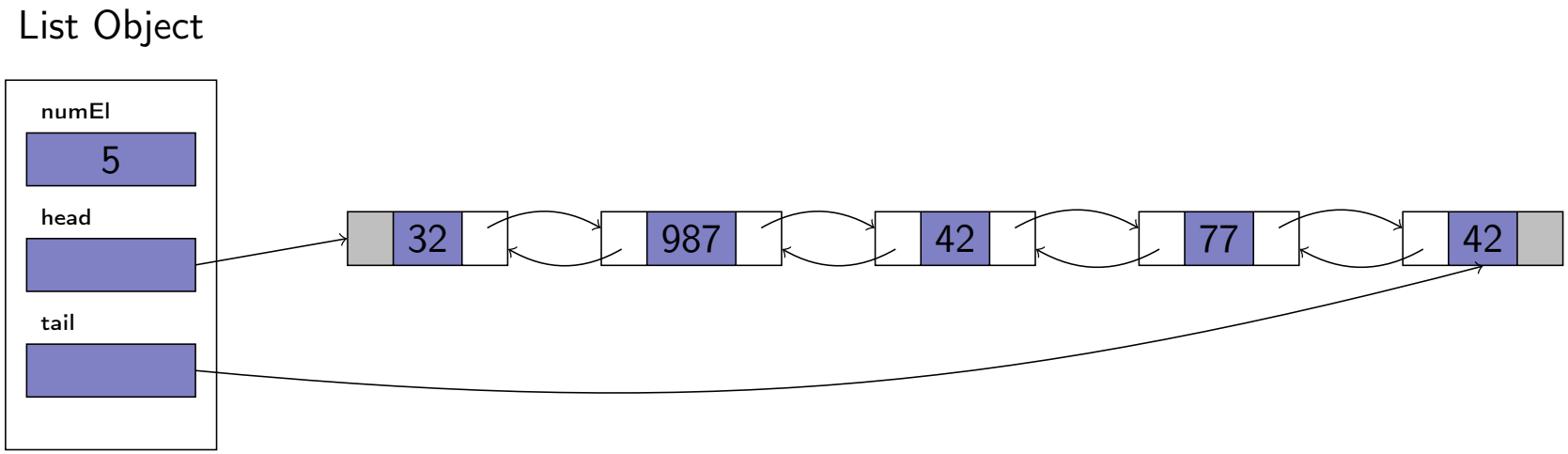
- What additional information do we need in our List and Node objects to support doubly-linked lists?
- What additional benefits are gained by using a doubly-linked node chain to implement a list?

Exercise 1



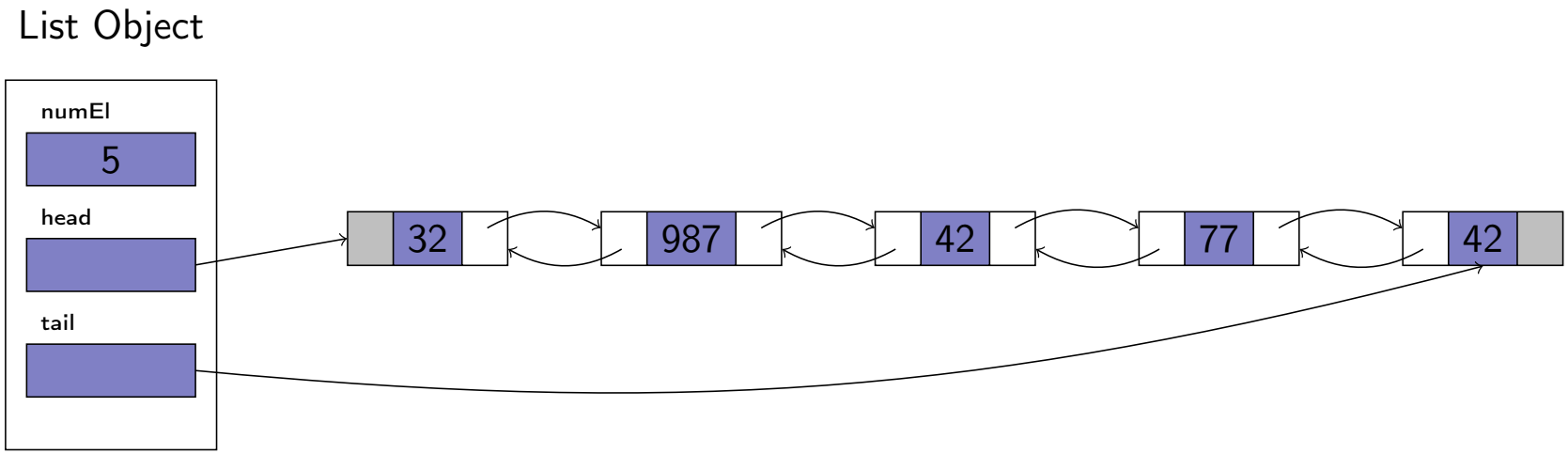
Draw this data structure after the first node is removed. Highlight the references that must be updated. Are there any scenarios where we might need to update the cursor?

Exercise 2



Draw this data structure after the last node is removed. Highlight the references that must be updated. Are there any scenarios where we might need to update the cursor?

Exercise 3



Draw this data structure after the middle node (42) is removed. Highlight the references that must be updated. Are there any scenarios where we might need to update the cursor?

Exercise 4

- Extend our `LinkedNode` class to be a doubly-linked node.

Exercise 5

- Extend our `LinkedList` class to be a doubly-linked list.

  (Just the class definition and additional instance variables for now)

- Problem: inherited `insertFirst()` method will create notes of type `LinkedNode<I>`, and we need doubly linked lists to have nodes of type `BilinkedNode<I>`. Solution?

Exercise 6

a) Implement the `insertLast()` method of the `BiLinkedList` class.

b) Implement the `deleteLast()` method of the `BiLinkedList` class.

c) Override the inherited `goLast()` method in `BiLinkedList` to improve it!

Exercise 7

- What should be our approach for adding additional cursor methods like `goBack()`?
- What classes or interfaces, if any, should be added and/or extended?