Midterm Review

Tiziana Ligorio
tligorio@hunter.cuny.edu

Basics

What should I study? Everything we covered up to and including Recursion

- Understand Classes/Objects, Constructors/Destructors, Templates
- Understand Inheritance/Polymorphism and their difference
- Understand the ADTs we covered and how their methods are implemented
- Know your projects
- Understand pointers, dynamic memory allocation/ deallocation
- Recursion

Basics

What should I study?

Everything we covered up to and including Recursion

What type of questions?

- writing C++ code
- understanding and answering questions about C++ code
- writing pseudocode
- answering questions about concepts

How do I prepare?

- read and understand the lecture slides
- supplement with book chapters or other resources (e.g. Prof. Weiss' notes)
- understand the programming projects
- review study questions on course webpage/schedule
- know well the ADTs we discussed and how their methods are implemented
- be prepared to implement operations from programming projects

Relax! (Not too much)

Don’t be intimidated

The exam will only ask about material we covered in class

You will not be asked to solve a problem you haven’t see before (perhaps a variation of something you have seen before)

There are no “trick questions”

If you understand the material covered in class and the projects you will do well!

Pseudocode

Code-like

Neat

All steps are there

All information necessary for computation is there

Omit implementation detail only

ex: `if n is a positive integer`

Think Algorithmically
Aka how to solve a problem

*"Experienced Computer Scientists analyze and solve computational problems at a level of **abstraction** that is beyond that of any particular programming language"*

Algorithm Design

- Identify the problem
- Come up with a procedure that will lead to solution
- Independent of implementation detail

Initial phase/step

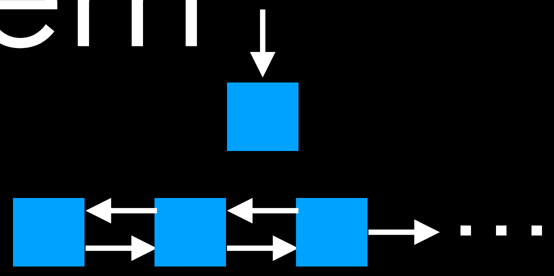
Model your problem/data

- **represent** the problem to support your algorithm

Implement solution

- Language
- Data structure
- Implementation detail

How to approach a problem



Instantiate new node

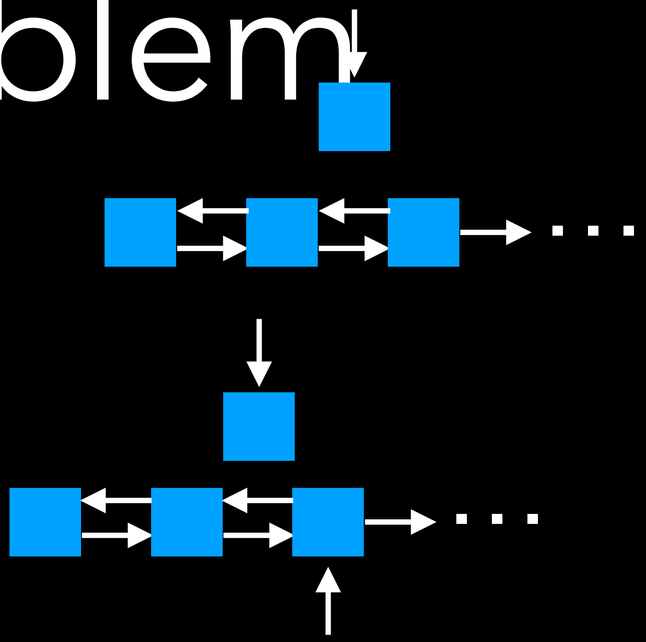
Obtain pointer

Connect new node to chain

Abstract thinking:
What are the steps?

Reconnect the relevant nodes

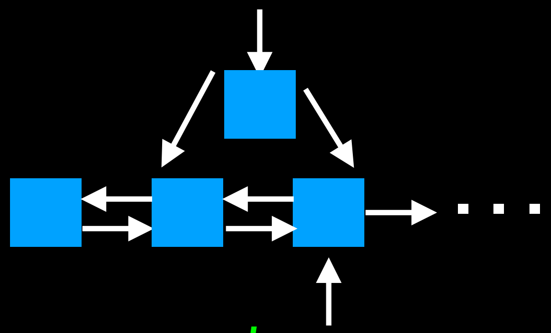
How to approach a problem



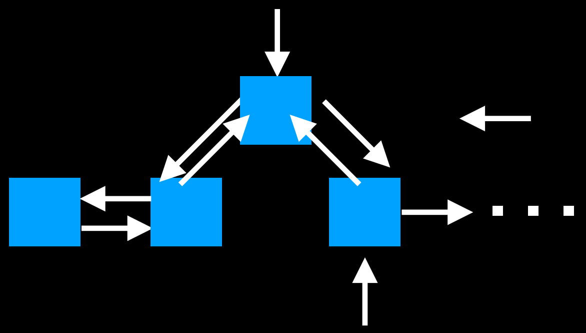
Instantiate new node

Obtain pointer

Connect new node to chain



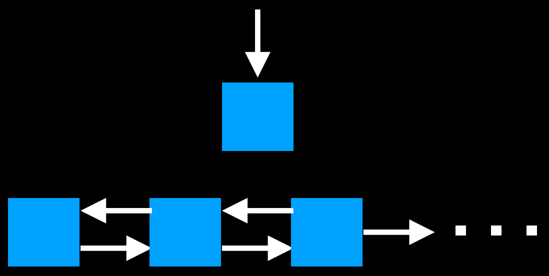
Reconnect the relevant nodes



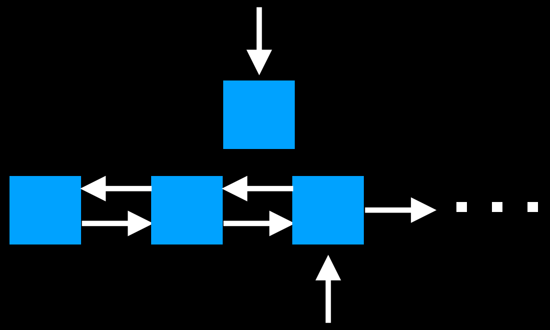
Draw Pictures

Start adding detail

Pseudocode

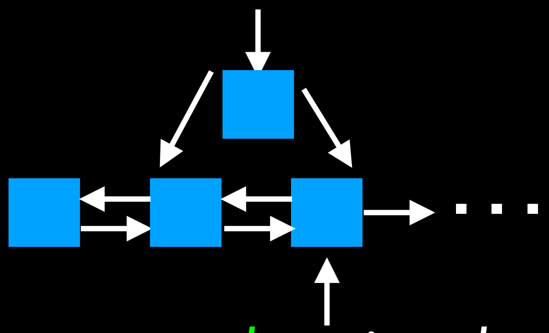


Instantiate new node to be inserted and set its value

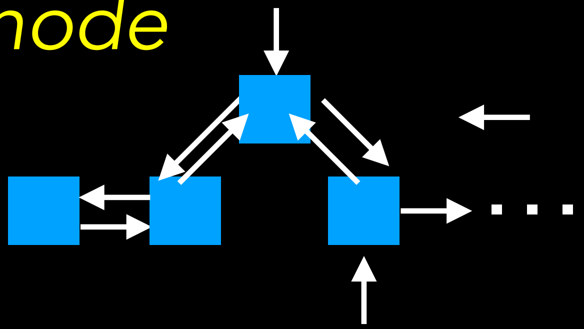


Obtain pointer to node currently at position 2

Connect new node to chain by pointing its next pointer to the node currently at position and its previous pointer to the node at position->previous

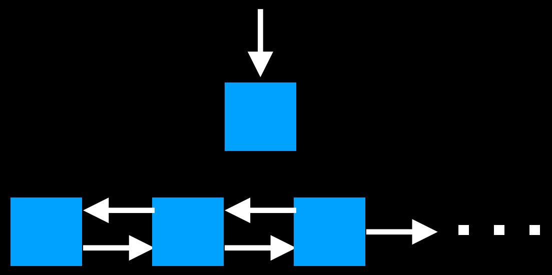


Reconnect the relevant nodes in the chain by pointing position->previous->next to the new node and position->previous to the new node

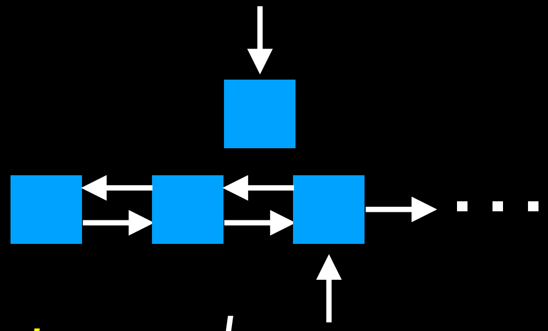


Order Matters!

More Pseudocodey

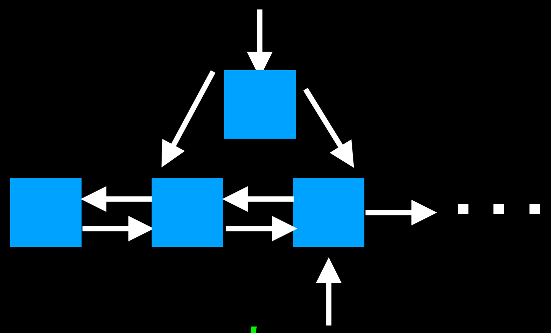


Instantiate new node new_ptr = new Node() and new_ptr->setItem()



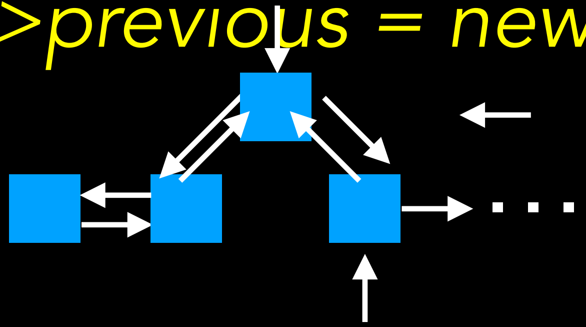
Obtain pointer position_ptr = getPointerTo(2)

Connect new node to chain new_ptr->next = position_ptr and new_ptr->previous = temp->previous

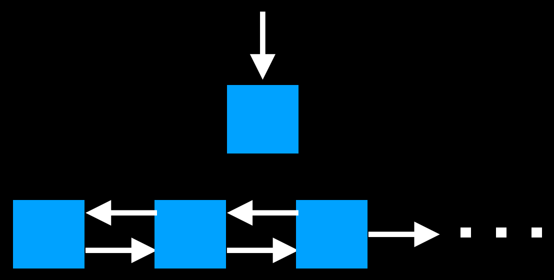


Reconnect the relevant nodes

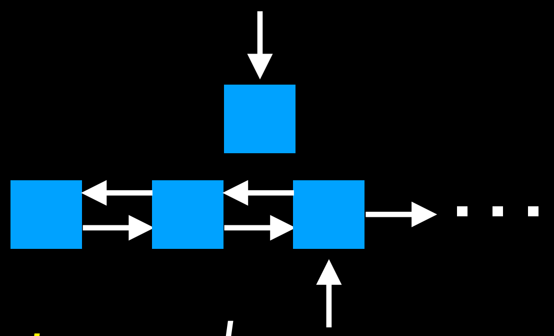
position_ptr->previous->next = new_ptr and position->previous = new_ptr



More Pseudocodey

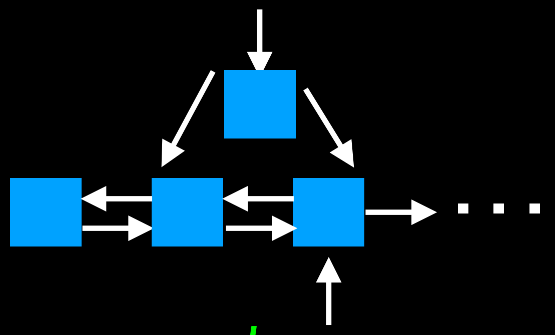


Instantiate new node new_ptr = new Node() and new_ptr->setItem()



Obtain pointer position_ptr = getPointerTo(2)

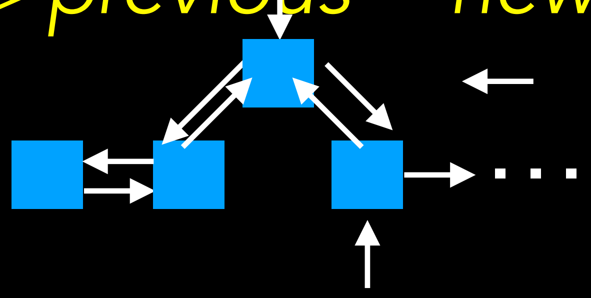
Connect new node to chain new_ptr->next = position_ptr and new_ptr->previous = temp->previous



Reconnect the relevant nodes

position_ptr->previous->next = new_ptr and

position->previous = new_ptr



If asked for code, translate to C++