# Exam Review 2

Chapter 5 – 9

CSC212 FG

CS Dept, CCNY

# Chapter 5: Linked Lists

- Node Class (Ex 1-4, 6-9)
- Linked List Toolkit (Ex 10 -16)
- The bag Class with a Linked List (Ex 19-21, 23,24,26)
- The sequence class with a Linked List (Ex 27-30: please refer to assignment 4)
- Comparison of dynamic arrays, linked lists and doubly linked lists (Ex 31-36)

# Chapter 6: Templates, Iterators and STL

- Template Functions and Template Classes
  - for code that is meant be reused in a variety of settings in a single program
  - Sections 6.1-6.2, Ex 1-3, 6-11
- Iterators (Sec. 6.5- 6.7, Ex 17-28)
  - step through all items of a container in a standard manner
- Standard Template Library (Section 6.3, Ex. 12-16)
  - the ANSI/ISO C++ Standard provides a variety of container classes in the STL

# All you need to know about Templates

- Template Function (Ex. 1-3)
  - a template prefix before the function implementation
  - template <class Item1, class Item2, ...>
- Function Prototype
  - a template prefix before the function prototypes
- Template Class (Ex. 6-10)
  - a template prefix right before the class definition
- Instantiation (Ex 11)
  - template functions/classes are instantiated when used

Better Understanding of classes and functions

# Iterators (Sec. 6.5- 6.7, Ex 17-28)

- We discussed how to build an iterator for the linked list
- so that each of the containers can build its own iterator(s) easily
- A node iterator is an object of the node_iterator class, and can step through the nodes of the linked list

# Linked List Version the bag Template Class with an Iterator

- Most of the implementation of this new bag is a straightforward translation of the bag in Chapter 5 that used an ordinary linked list
- Two new features
  - Template class with a underlying type Item
  - iterator and const_iterator – defined from node_iterator and const_node_iterator, but use the C++ standard [...) left inclusive pattern

# Standard Template Library (STL)

- The ANSI/ISO C++ Standard provides a variety of container classes in the STL
  - set, multiset, stack, queue, string, vector
- Featured templates and iterators
- For example, the multiset template class is similar to our bag template class
- More classes summarized in Appendix H

# Chapters 7/8 Stacks and Queues

- Stacks and LIFO(Read Chapter 7, esp. 7.1 and 7.3)
  - Self-Test: 1, 2, 7, 8, 9, 10,
- Queues and FIFO (Read Chapter 8, esp. 8.1 and 8.3)
  - Self-Test: 1, 2, 6,10, 11,12
- Priority Queues (Read Section 8.4)
  - Self-Test: 16, 17
- References Return Values (Read Section 8.5 and p. 302 in Chapter 6)
  - Self-Test: class note of Lecture 12

# Chapter 9 Recursive Thinking

- Recursive Functions (Section 9.1)
  - Recursive Calls and Stopping Cases
  - Activation Record and Run-Time Stack
  - Self-Test: Exercises 1-4
- Reasoning about Recursion (Section 9.3)
  - Infinite Recursion and One Level Recursion
  - Ensure no Infinite and Correctness
  - Ex. 9, 10 on page 448

- Applications of Recursion (Optional :Section 9.2)

# A quiz on queue

- Implement
  - queue
  - push
  - pop
  - front
- Check pre-conditions if needed

```
template <class Item>
  class queue
  {
  public:
    // TYPEDEFS and MEMBER CONSTANTS
    typedef std::size_t size_type;
    typedef Item value_type;
    static const size_type CAPACITY = 30;
    // CONSTRUCTOR
    queue( );
    // MODIFICATION MEMBER FUNCTIONS
    void push(const Item& entry);
    void pop( );
    // CONSTANT MEMBER FUNCTIONS
   Item front( ) const;
   bool empty( ) const { return (count == 0); }
    size_type size( ) const { return count; }
  private:
    Item data[CAPACITY];    // Circular array
    size_type first;        // Index of item at front of the queue
    size_type last;         // Index of item at rear of the queue
    size_type count;        // Total number of items in the queue
    // HELPER MEMBER FUNCTION
    size_type next_index(size_type i) const
            { return (i+1) % CAPACITY; }
  };
```

# A few suggestions for preparation

- Make sure you understand each line of your assignments 4 (including node.cxx).
- Make sure you understand the two implementation files for both stack and queue.
- Key concepts: circular array; front_ptr/rear_ptr for queue; precuror; template classes/functions
- Exercise recursive thinking, e.g. with Assignment 5; be able to write recursive code for small problem (or printing digits vertically example code)
- Be able to write code with queue/stack for small problem(parenthesis balancing code);

# Exam 2

- Nov 07, 2016, from 4:00 to 5:30 pm