Exam I Review

CS 3358
Spring 2015

Jill Seaman

Exam I

- Monday, March 2
- In class, closed book, closed notes, clean desk
- 20% of your final grade
- 80 minutes
- I recommend using a pencil (and eraser)
- All writing will be done on the test paper I will hand out.
- No calculators or cell phones or head phones.

Exam Format

- 100 points total
  - Writing programs/functions/code (about 50%)
  - Multiple choice
  - Fill-in-the-blank/short answer
  - Tracing code
    - what is the output OR
    - show the diagram of a linked list
  - Finding errors in code (maybe)

Arrays, pointers, structs

- Data types, scalar, composite (arrays)
- First-class vs second-class objects
- Pointers: declare, assign, use (dereference)
- Dynamic memory allocation (and deallocation)
- Structures, pointers to structures, objects (–>)
- Shallow copy vs. deep copy

Objects and classes

- Encapsulation, Information hiding, Interface
- Class declaration
  - data members, member functions, public and private
- Default parameters, initializer list, const member function
- The big three (defaults, when to override)
  - destructor, copy constructor, operator=
- Operator overloading
- How to separate source code into multiple files
- Know how to implement Card/Deck/Player

Linked Lists

Read chapter 17 in Gaddis, NOT in Weiss book.

- How to define a linked list
  - Node definition (next, previous)
  - head (tail, ...)
- Using null pointers

Demo code is on the class website.

- Basic operations: be able to implement for single or doubly linked list. (NumberList demo)
  - constructor, append, insert, remove, destroy
  - display the list, copy constructor
- Know how to draw the lists from code
- Arrays vs. linked lists: pros+cons

Introduction to ADTs

- Data structure vs abstract data type (definitions)
- Commonly used ADTs (list, set, bag, map)
  - understand the operations, be able to implement
- Implementation vs. interface of an ADT
  - abstract and concrete parts of the implementation
- bag implementations:
  - version 1: fixed length array
  - version 2: dynamically allocated array
    - how to resize a dynamic array
- List_3358 demo and PA2 (arrays and linked lists)

Introduction to C++ STL

- containers vs iterators
- Know how to use vectors:
  - operations described in the slides only:
  - W1 Arrays, Pointers, Structs, slides 9 and 10
  - W5 ADT Intro, slides 27, 28, 29
- Be able to read code that uses an iterator

Analysis of algorithms

- Understand the concept:
  approximating the amount of time it takes to execute an algorithm by counting statements, in terms of data size (N).
- Know the growth rate functions
  - Which ones are faster growing than others
- For a given algorithm/code sample, be able to determine the Big O function (to say it is O(**F(N)**))
- Given two implementations, be able to say which is more efficient (faster) than the other, based on their Big O functions.

Example Programming Problem

Given the class declaration (provided in the test) for a bag implemented as a singly-linked list, write C++ code to implement member functions that will

a) add an item to the bag.

b) return the number of occurrences of a given element in the bag.

You should be prepared to implement the ADTs whose operations were described in the ADT Intro lecture, using array, dynamic arrays, or linked lists.

Example Tracing Problem

Draw a picture to depict the nodes in memory after the following code is executed.

```
struct Node {
    int data;
    Node *next;
    Node *other;
};

…
Node *ptr;
Node *temp = new Node;
temp->data = 42;
temp->other = temp;
temp->next = NULL;
ptr = temp;

temp = new Node;
temp->data = 13;
temp->next = ptr;
```


Example Short Answer

What is the Big O function for the insert operation in a doubly linked list when inserting before the cursor?

I will provide the code for the operation this time

Answer would be something like: O(n) or O(1) or O(n^2) ...

Practice: figure out the Big O functions for all of the operations in the ADT implementations in the slides and demos and programming assignments.

How to Study

- Review the slides
  - understand all the concepts, quiz yourself!
- Use the book(s) to help understand the slides
  - there will be no questions over material (or code) that is in the book but not on the slides
- Understand the code in the demo(s)!
- Understand the programming assignments
  - rewrite yours so they work correctly!
- Practice, practice, practice
- Get some sleep