Exam I Review

CS 3358
Summer II 2013

Jill Seaman

Exam I

- Friday, July 19, 12:00pm to 1:30pm
- Derr 241 (here)
- Closed book, closed notes, clean desk
- 20% of your final grade
- I recommend using a pencil (and eraser)
- All writing will be done on the test paper I will hand out.
- No calculators.

Exam Format

- 100 points total
  - Writing programs/functions/code (at least 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

- First-class vs second-class objects (types)
- Know how to use vectors
  - just the operations in the lecture slides, no iterators
- 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
- 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
- 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
- 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)

Analysis of algorithms

- Understand the concept: approximating 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/function, be able to come up with 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.

Objectives covered

- Write C++ programs using multiple classes and arrays of objects.
- Read and write C++ code that uses pointer variables and memory operations (new, &, *, delete), pointers to arrays, structures, and objects and the -> operator.
- Write C++ programs with the source code separated into multiple files, using header (.h) files.
- State the definition of an Abstract Data Type.
- Describe the semantics (values and operations) of the following ADTs : Lists, Sets, Multi-set, Map
- Perform (demonstrate) each of the List ADT operations given an instance of the ADT.
- Summarize the advantages and disadvantages of array vs linked lists
- Implement the List (Set, Multi-set, Map) ADT in C++ using arrays and linked lists
- Write programs in C++ that use vector from the C++ Standard Template Library
- List (in order of increasing growth rate) the 6 categories of functions used in analyzing algorithms.
- Analyze algorithms, including implementations of ADT operations, for efficiency (give the big O function)

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.

The class declaration will include the prototypes for the member functions. It may or may not include the member variable definitions.

It could be a bag or a set or a list (direct access or cursor based) or a map, and some appropriate operations.

I could ask you to implement it using an array, a dynamic array, or a linked list

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 *foo;
};

…
Node *hey;
Node *temp = new Node;
temp->data = 42;
temp->foo = temp;
temp->next = NULL;
hey = temp;

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

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 list implementations.

How to Study

- Review the slides
  - understand all the concepts
- 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