Exam I Review

CS 3358
Summer I 2012

Jill Seaman

Exam I

- Thursday, June 14, 5:40pm to 7:00pm
- 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 (~50%)
  - Multiple choice
  - Fill-in-the-blank/short answer
  - Tracing code (what is the output)
  - Finding errors in code

Arrays, pointers, structs

- First-class vs second-class objects (types)
- Know how to use vectors
  - just the operations in the slides, no iterators
- Pointers
- Dynamic memory allocation (and deallocation)
- Structures, pointers to structures
- 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

Introduction to ADTs

- Data structure vs abstract data type
- Commonly used ADTs (list, set, bag, ...)
- Implementation vs. interface
- bag implementations:
  - version 1: fixed length array
  - version 2: dynamically allocated array
- List_3358 demo

Linked Lists

- How to define a linked list
  - Node definition
  - head (tail, cursor, ...)
- Using null pointers
- Basic operations: be able to implement for single or doubly linked list.
  - constructor, append, insert, remove, destroy
  - display the list, copy constructor
- Know how to draw the lists
- Arrays vs. linked lists: pros+cons

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.

Example Programming Problem

Given the included header file for a bag implemented as a singly-linked list, write member a function that will

a) add an item to the bag.

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

I would provide you with a header file (class declaration) probably including the prototypes for the member functions.

Example Tracing Problem

Draw a picture to depict the nodes in member 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

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 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 homework assignment solutions
  - rewrite yours so it works (solutions on TRACS)
- Practice, practice, practice
- Get some sleep