Exam 2 Review

CS 2308
Fall 2018

Jill Seaman

Exam 2

- Monday, November 19
- In class, closed book, closed notes, clean desk
- 15% of your final grade
- 80 minutes to complete it
- Bring your ID card!
- Bring a number 2 pencil and eraser.
- NO: calculators or cell phones.
- NO: headphones/earbuds.

Exam Format

- 100 points total
  - 50 pts: 25 Multiple choice/matching (scantron form)
  - 50 pts: Writing functions/code,
    how to set up a multi-file project in linux
- Tasks:
  - Tracing code (what is the output)
  - Finding errors in code
  - Demonstrate general knowledge about C++ and programming
  - Programming (writing code, like in the PAs)

Content from Textbook

- Intro to Classes (Unit 4), (13.1-12)
- Multi-file Development (from the Lecture slides)
- Pointers to Structures (11.9)
- Linked Lists (Unit 5) (18.1-2)

Unit 4: Intro to Classes

- Declaring a class:
  - Members: variables and functions
  - private vs public, access rules
  - class declaration, member function definitions
  - accessor/mutator (getter/setter), const
  - defining objects and accessing members
- Procedural programming
- Object oriented programming:
  - **encapsulation**
  - **data (or information) hiding**
  - **interface**

Unit 4: Intro to Classes (cont)

- Inline member functions
- Constructors
  - How to name, return type?
  - When are they called? what do they do?
  - Default constructor
  - passing arguments to constructors
- Destructors
  - what it is, how to name it, when is it called?
- Overloaded constructors
- Arrays of objects: declare, initialize, and access

Multi-file development on Linux

- How to split up a program with classes
- Header files vs .cpp files, when to include which
- How to compile multiple file program:
  - using g++ :
    $ g++ a.cpp b.cpp
  - separate compilation:
    $ g++ -c a.cpp
    $ g++ -c b.cpp
    $ g++ a.o b.o
  - makefile:
    understand the example, basic rules, how to make

Pointers to Structs

- How to declare and assign pointers
- Structure pointer operator: ->
- How to access members through the pointer
  - `(*p).member vs. p->member vs. *(p.member)`
- dynamic allocation of structures

Unit 5: Linked Lists

- Dynamically allocated list data structure
- Organization: nodes, head pointer, empty list, NULL
- Linked list tasks: T1-T11:
  - create empty list, create a new node
  - add to front of list
  - append to end of non-empty list
  - traversing a linked list (display, count, sum, etc)
  - how to advance 2 pointers together (n and p)
  - delete given n and p, special cases
  - insert given n and p, special cases
  - linked list destruction
- NumberList class

Sample Problem: multiple choice

You want to make p point to the node containing num, and make n point to the node previous to the one containing num. What is wrong with this loop?

```
ListNode *n, *p=head;
while (p->value!=num) {
  n = p;
  p = p->next;
}
```

(a) The 2 statements inside the while are in the wrong order.
(b) If num is not in the list, p will become NULL, and p->value will cause an error.
(c) If num is not in the list, it will be an infinite loop.
(d) After the loop both n and p will point to the node containing num.
(e) There is nothing wrong with the loop.

Sample Programming Problem

Consider a Rectangle class, which stores a rectangle using two floating point values: width and height.
It has 2 constructors (one default, one with 2 arguments), and set and get function for the width and height.
The default rectangle has width and length equal to 1. The other constructor takes the initial width and height.
There is a function print() that outputs the width and height.
The rectangle also has a function area() that calculates and returns the area of the rectangle.
(a) Write the class declaration.
(b) Implement all the class functions (do not inline the definitions)

Sample Programming Problem

Write C++ statements to perform the following tasks, given this definition of a linked list:

```
struct Node {
   int data;
   Node *next;
};
Node *head = NULL;
```

- **add to front:** add a new node with value 10 to as the first node of the list. The list may or may not be empty before the node is added.
- **sum:** output the sum of all the numbers in the list. Your code should work for a list of any length.

How to Study

- Review the slides and Squarecap questions
  - understand all the concepts, quiz yourself
- 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
- Review programming assignments (fix yours!)
  - get printouts of solutions in my office
- Do the practice exercises.
- Practice, practice, practice! Write code!
- Get some sleep