# CS 202 Computer Science II
# Spring 2016 Syllabus and Information

**Table of Contents: table of contents, what we will do today, assignments**
**Instructor Directories:** cd ~jkinne/public_html/cs202-s2016/
**Course website and information:**
https://docs.google.com/document/d/1QadsJAAQgscX6I13gDF6HIoHY3-hX2AX_6oa0TwSs3E/edit?usp=sharing

## Table of Contents

**table of contents, what we will do today, assignments**

## General Information

### Contact Your Instructor

**Name:** Jeff Kinne

**Email:** jkinne@cs.indstate.edu
**Phone:** 812-237-2136
**Office:** Root Hall, room A-129

## Lecture, Exam, Office Hours

**Lecture:** Tuesdays and Thursdays from 11:00-12:15pm in Root Hall, room A-017.

**Exam:** Thursday, May 5, 10-11:50am in A-017.

**Instructor Office Hours:** I am generally in my office and available most MWF's from about 8:30am-4pm. My official office hours are Wednesdays 9:30-11:30am.

**GA Tutoring**: We have a few graduate assistants who will be in the computer science unix lab, room A-015 in the basement of Root Hall, for about 20 hours per week in total. You can go to this lab to work on your programs. The computers are unix machines, and you can use the cs202xx login that will be sent to you during the first week of class to use them. Or, you can bring your laptop to work on. Either way, you can ask the graduate assistants to look at your programs, and you can work with any other CS 202 students that are there (you could use the lab as a regular meeting place to work with your classmates). The regular hours that the lab will be open will be posted to **here on the department's website**.

**Website**: this google doc, or find a link from kinnejeff.com

## Prerequisites

CS 201 with a C or better.

## Recommended text

There is no required text.  The following will be used as references

- CS 202 as taught by Geoff Exoo, Exoo's CS 201/202 Outline
- CS 201 as taught by Jeff Kinne
- The C Book
- C Programming Wikibook
- How to Think Like a Computer Scientist C++ Verson
- The C Programming Language by Kernighan and Ritchie
- cplusplus.com
- MIT course - Practical Programming in C
- ACM Programming Contest Problems

### Course Announcements

Announcements regarding the course will be made both during class and via email to your @sycamores.indstate.edu email address. You should regularly check this email account or have it forwarded to an account that you check regularly. You can set the account to forward by logging into your indstate.edu email from Internet Explorer (the "light" version of the webmail client that opens up from Firefox or Chrome does not give the option to forward email).

### Classroom conduct

You may not use cell phones, iPods/music players, etc. during class. You should be civil and respectful to both the instructor and your classmates, and you should arrive to class a few minutes before the scheduled lecture so you are ready for lecture to begin on time. You may use your computer during class if you are using it to follow along with the examples that are being discussed. You may not check email, facebook, work on other courses, etc. during class.

## Course Description

The official description of this course from the catalog is

“This course is a continuation of CS 201. It involves a deeper study of programming languages, but emphasizes programming in a particular language. Topics include algorithm design and analysis, data structures, recursion, threads, network programming, graphics, security, and ethics.”

In brief, this course completes your introduction to C/C++ programming and data structures and algorithms (together with CS 151, 201, and 303). The course will be fairly heavy in programming, but we start to work on more and more non-trivial, interesting programs (that are more efficient than the program you would first think of).

## Course Outline

We begin the course by following along with the outline for CS 201/202 put together by Geoff Exoo that is linked above. Along the way, we will get a view of each of the following topics.

1. C programming refresher with some interesting example and ICPC problems
2. C programming - anything you haven’t seen yet?
    a. Review of all the keywords and symbols, what haven’t you seen
    b. Review of commonly used libraries (I/O, math, string, time/date, …)
3. Editors, command-line tools, etc. - things that are useful for developers
    a. Editors - vim, emacs

   b. Command-line tools - man
4. C++ programming - object-oriented programming and C++ syntax
5. Basic algorithms/techniques
   a. Brute force
   b. Divide and conquer
   c. Dynamic programming
   d. Graph algorithms
6. Running time analysis of algorithms - big O and such, recursion trees
7. Basic data structures
   a. Arrays (fixed size, and that can grow)
   b. Linked Lists (and queues, stacks)
   c. Binary trees (unbalanced and balanced)
   d. Hash tables
   e. Heaps (aka priority queues)
   f. Adjacency matrix, adjacency list
   g. Maybe some others
   h. C++ versions of basic data structures
8. Something interesting and fun (network programming, graphics, threads, extremely large number arithmetic, …)
9. Maybe - compiling/designing programs in IDEs (Eclipse, MS Visual Studio)

## Grading and Assignments

The students of this course have the following responsibilities: read assigned readings before lecture, attend lecture, complete homework assignments, take in-class quizzes, take exams, and complete a project. The final grade consists of:

- **Project: 15%** of the final grade.

- **Homeworks and Quizzes: 30% total**. Most weeks there will be at least one homework assignment or quiz.

- **Exams: 45% total**. There will be 3 exams. The total exam grade will be calculated as
  max( (.1 * exam1 + .15 * exam2 + .2 * exam3) / .45,
  (.15 * exam2 + .2 * exam3) / .35,
  exam3)

- **Class Attendance/Participation: 10% total**. Attendance will be taken at the beginning of each class. Half of your attendance/participation score will consist solely of whether you were present when attendance was taken each day - the total number of days present divided by the number of lectures in the semester. The other half of your attendance/participation grade will be assigned at the end of the semester based on how attentive you were in class throughout the semester.

## Late Homeworks

All homework assignments will be given a preferred due date. Assignments can be turned in past the preferred due date, but any assignments turned in late will have their value multiplied by 80% (so the highest grade you can get on a late assignment is 80%). Each assignment will have a “final due date” past which no credit will be given.

## Start Homeworks Early

I suggest attempting a homework assignment the day it is given, or the day after, so that if you have a problem you can ask early. If you continue to have problems in trying to complete the assignment, you will have time to ask again. Many of the homework assignments require thought and problem solving, which takes “time on the calendar” not just “time on the clock”. By that I mean that spending an hour on 3 consecutive days is likely to be more productive than trying to spend 3 hours at once on the assignment.

## Expected Amount of Work

My expectation is that an average student will spend about 4-6 hours OUTSIDE of class each week (that is in addition to class time) WORKING PRODUCTIVELY/EFFICIENTLY (not just staring at the computer) to complete their coursework for this class. Some students may spend less time than this, and some students will spend more.

**Note - this is your most important class, by far (for CS majors). Also, your classes should be more important than your part-time job.**

## Grade Cutoffs

I will design homework assignments and exams so that a standard cutoff for grades will be close to what you deserve. After the first exam I will create a grade in Blackboard called “Letter Grade” that is what your letter grade would be if the semester ended today. Initially, I will assign the following grades: 93-100 A, 90-93 A-, 87-90 B+, 83-87 B, 80-83 B-, 77-80 C+, 73-77 C, 70-73 C-, 67-70 D+, 63-67 D, 60-63 D-, 0-60 F

My goal is that the different grades have the following rough meaning.

**A+/A**
You understand everything and probably could teach the course yourself.

**B+/A-**
You understand nearly everything, and should be all set to use this knowledge in other courses or in a job.

**C/C+/B-/B**
Some things you understand very well and others you don't (more towards the former for a B and more towards the latter for a C).

**D-/D+/C-**
You did put some effort in, and understand many things at a high level, but you haven't mastered the details well enough to be able to use this knowledge in the future.

**F**
Normally, students that get an F simply stopped doing the required work at some point.

### Blackboard

The course has a blackboard site. Click here to go to blackboard. You should see this course listed under your courses for the current term. The blackboard site is only used for giving you your grades (go to the course in blackboard, then click "My Tools", and then "My Grades"). All course content, schedule, etc. is kept in this google doc (which you are currently viewing).

## Academic Integrity

Please follow these guidelines to avoid problems with academic misconduct in this course:

- **Homeworks:** You may discuss the homework assignments, but should solve and finish them on your own. To make sure you are not violating this, if you discuss with someone, you should DESTROY any work or evidence of the discussion, go your separate ways, SPEND at least an hour doing something completely unrelated to the assignment, and then you should be able to RECREATE the program/solution on your own, then turn that in. If you cannot recreate the solution on your own, then it is not your work, and you should not turn it in.

- **Note on sources:** if you use some other source, the web or whatever, you better cite it! Not doing so is plagiarism.

- **Exams:** This should be clear - no cheating during exams. The exams will be closed-book, closed-notes, no computer, and no calculator.

- **Projects:** You should not copy from the internet or anywhere else. The project should be your own work. It will be fairly obvious to me if you do copy code from the internet, and the consequences will be at the least a 0 on the project.

If cheating is observed, you will at the least receive a 0 for the assignment (and may receive an F for the course), and I will file a Notification of Academic Integrity Violation Report with Student Judicial Programs, as required by the university's policy on Academic Integrity. A student who is caught cheating twice (whether in a single course or different courses) is likely to be brought before the All-University Court hearing panel, which can impose sanctions up to and including suspension/expulsion. See the Student Code of Conduct and Academic Integrity Resources for more information.

Please ask the instructor if you have doubts about what is considered cheating in this course.

## Special Needs

If you have special needs for the classroom environment, homeworks, or quizzes, please inform the instructor during the first week of classes. If you have any such needs, you should go to the Student Academic Services Center to coordinate this. See Student Academic Services Center - Disabled Student Services for more information.

## Disclosures Regarding Sexual Misconduct

Indiana State University fosters a campus free of sexual misconduct including sexual harassment, sexual violence, intimate partner violence, and stalking and/or any form of sex or gender discrimination.  If you disclose a potential violation of the sexual misconduct policy I will need to notify the Title IX Coordinator.  Students who have experienced sexual misconduct are encouraged to contact confidential resources listed below.  To make a report or the Title IX Coordinator, visit the Equal Opportunity and Title IX website: http://www.indstate.edu/equalopportunity-titleix/titleix.

The ISU Student Counseling Center – HMSU 7th Floor | 812-237-3939 |  www.indstate.edu/cns
The ISU Victim Advocate – Trista Gibbons, trista.gibbons@indstate.edu
HMSU 7th Floor | 812-237-3939 (office)  |  812-230-3803 (cell)

**table of contents, what we will do today, assignments**

Campus Ministries - United Campus Ministries | 812-232-0186
http://www2.indstate.edu/sao/campusinistries.htm
www.unitedcampusministries.org | ucmminister2@gmail.com
321 N 7th St., Terre Haute, IN  47807

For more information on your rights and available resources
http://www.indstate.edu/equalopportunity-titleix/titleix

## Assignments

Assignments will be posted to this document and announced in class. Things will be listed here most recent first.

### Reading/Viewing Assignments

Check “What will we do today” for reading assignments.

### Exams

#### Exam the first

- March 1 in class
- topics - everything we’ve covered. All syntax in C.
- half on paper, half on computer.
- Question types
    - C program, and what is the output, or values of variables, or how many times a loop ran, or how much memory is used, or something else.
        - some will be like we’d have in a real program - kind of obvious or self-explanatory
        - some will be strange - only way figure out is to it step by step.
        - int *x, y, *z, w[10];
          x = &y; w[5] = 3; z = x; *z = 6; y = 7; *x = 11; …
          maybe things get passed into functions and possibly modified there.
    - Complete a C program - I give you most, you finish it.
    - C program from scratch.
    - Use one of the data structure files we’ve created to do something.
    - Math - logs, powers, sums, hex/bin/decimal, big O
    - Data types - binary, 1’s complement, 2’s complement
    - Data structures - the ones above.
    - Correct solutions from HW assignments might be relevant.
- Actual exam is at - http://cs.indstate.edu/~jkinne/cs202-s2016/cs202-exam1.txt
- Grading
    - Paper
        - See exam itself for some grading information.
        - Additional notes…
            - 2a. -1 if all correct except just for a single operation (not for I of them), -.5 if had at least a few of them correct, -1.5 if had at least something that wasn’t nonsense.

        - 2b.  -.5 for each that is wrong.  -.5 total for the problem if didn't include anything about "why".
        - 
    - Computer
        - Each program is graded out of 1, and then weighted as 6, 3, or 1, for a total of 10 possible points.  Each program gets full credit (exactly correct, or only minor typo in output), 70% (on the right track, some silly mistake in code), 40% (compiles, does something), or 0% (doesn't compile or isn't at all on the right track).

## Project

- 
- Grading
    - 10 points for correctness and functionality - did you implement everything I suggested, and how well does it work?
    - 3 points for commenting and style.  You should use good variable/function names, break code up into functions that are not more than a screen long, include comments for each function and loop, have comments at the top of the file saying what it does, have code that is properly indented and has blank lines in a way that makes it easy to read.
    - 2 points for error-checking - make sure the program works regardless of what the user types (there should be no way for the user to break the program).  If the user must pay attention to case, let them know that, and deal with it appropriately if they use the wrong case.
-

## In-class quizzes

These may be given roughly once per week at the beginning, and may or may not be announced.  You will generally be asked either a relatively simple question about what was covered in the last class, or you will be asked an extremely simple question about the reading/viewing assignment for that day.  The in-class quizzes will count under the category of "homework" in the grading breakdown in the syllabus.  Questions on quizzes are normally graded as either correct, incorrect, or half credit.  There will be NO makeups for quizzes; if you have an excused absence the quiz simply will not count as anything.

### Quiz 5

- Date: 4/12
- HW points:  4.  Note, will count, max(q4, q5).

**table of contents, what we will do today, assignments**

- You'll have: 10 minutes
- See quiz5.txt in the usual place

## Quiz 4

- Date: 3/31
- HW points:  4.  Note, will count, max(q4, q5).
- You'll have: 10 minutes
- See quiz4.txt in the usual place

## Quiz 3

- Date: 2/9
- HW points:  Note, will count, max(q2, q3).
- You'll have: 15 minutes
- LAST NAME, FIRST NAME
- (1) 1/2 point.  Simplify the following expression, where ^ is superscript and _ is subscript.
  $(10^5)^4 * \log_{10}(10^3*10^2) / 1000$
  $10^2 * \log_{10}(10^5) / 10^3 = 5/10 = 1/2$
- (2) 1/2 point.  Give a simple formula for the following -
  20 + 40 + ... + (20*55)
  20 (1 + 2 + … + 55) = 20 * 55 * 56 /2
- data structures: unsorted array, sorted array, linked list, balanced binary search tree, hash table
- 
- (3) 1.5 points.  for data structure with worst average-case running time for insert, what are the average-case running times for insert, lookup, delete, and 1-2 sentence description of insert.
  - sorted array
- (4) 1.5 points.  for data structure with best average-case running time for lookup, what are the average-case running times for insert, lookup, delete, and 1-2 sentence description of insert.
  - hash table

## Quiz 2

- Date: 2/1
- HW points:  Note, will count, max(q2, q3).
- You'll have: 15 minutes
- LAST NAME, FIRST NAME
- (1) ½ point - $8 * \log_2(2^5 * 2^4)$ - simplify, okay to write as power
  - Answer: 72
- (2) ½ point - 5 + 10 + 15 + … 100 = (okay to write a formula)
  - Answer: 5 * (1 + … + 20) = 5*20*21/2 = 1050

- 
- unsorted array, sorted array, linked list, balanced binary search tree, hash table
- 
- (3) 1.5 points - for data structure with best average-case insertion time, what are the average-case running times for insert, lookup, delete, and 1-2 sentence description of insert.
    - hash table or unsorted array
    - grading: ½ point for picking the right one, ½ point for running time, ½ point for description.
- (4) 1.5 points - for data structure with worst average-case delete time, what are the average-case running times for insert, lookup, delete, and 1-2 sentence description of insert.
    - any of unsorted array, sorted array, linked list.
    - grading - same as #3

## Quiz 1

- Date: 1/19
- HW points: 4 points.  Note, will count, max(q0, q1).
- You'll have: 15 minutes
- LAST NAME, FIRST NAME
- (1) *Rules of exponents with logarithms and exponentials.  Right or wrong.*
    - simplify $(\log_{10}(100^2) * 4)^{½}$
    - Answer: 4
- (2) *Converting between binary, decimal, hex. right, half, or wrong.*
    - Convert 3f from hexadecimal to binary and decimal
    - Answer: 11 1111 in binary, 63 in decimal
- (3) *What does C program output that has if's, loops, functions.  I will try to trick you. Right, wrong, or half*
    - What is the final value of x?
    - char *s = "this is good";
    - inx x = 0;
    - for(int i = 0; s[i] != '\0'; i++) {
    -   if (isalpha(s[i]) x++);
    -   else x--;
    - }
    - Answer: 8, or accept - syntax error with ) in if line
- (4) *Also simple C statement(s) with some operations (arithmetic, bit ops, boolean ops).  I will try to trick you.  Right, wrong, or half*
    - What is the final value of x?
    - char x = 127;
    - x &= 7; // x=7

- x = x % 5; // x=2
- x = x * (3.5/2); // x = 2 * (3.5/2) = 2 * (1.75) = (int) 3.5) = 3
- Answer: 3

## Quiz 0

- Date: 1/14
- HW points: 4 points
- You'll have: 15 minutes
- LAST NAME, FIRST NAME
- One question each on…
  - (1) *Rules of exponents with logarithms and exponentials. Right or wrong.*
    - simplify $(\log_2(256))^{½} * 8$, give an integer answer or power of 2
    - $2^{9/2}$
  - (2) *Converting between binary, decimal, hex. right, half, or wrong.*
    - Convert 97 from base ten to binary and hex
    - 0110 0001 in binary, 61 in hex
  - (3) *What does C program output that has if's, loops, functions. I will try to trick you. Right, wrong, or half*
    - What is the final value of x?

    -
      ```
      char *s = "hello";
      int x = 1;
      for(int i=0; i < strlen(s); i++) {
        if (islower(s[i])) x += x; else x /= 2;
      }
      ```
    - final answer, x = 32
  - (4) *Also simple C statement(s) with some operations (arithmetic, bit ops, boolean ops). I will try to trick you. Right, wrong, or half*

        - What is the final value of x?
          float f = 3.14159;
          int i = f * 2;
          int x = ((i / 3) << 1) | 7;
        - final answer, x = 7
-

## Ungraded Assignments

Things you should do, but I won't grade and probably won't go over in class. Note that we only have time in class to talk about maybe 70% of what you should know.

## Graded Assignments

### HW 10

- Due 4/26
- Late due date: 4/28
- Graphs - take graphs_hw10.cpp, call it ~/handin/hw10.cpp
    - 0) create some example graph, call it hw10_SOMETHING_DESCRIPTIVE.txt. Include a short description (could be just a few words) at the end, where it won't mess up the program.
    - a) add DFS. Instead of queue, use stack.
    - b) add a method that checks if the graph is connected (run BFS, and then check for INT_MAX distances).
    - c) add a method that checks if the graph has a cycle. Use the BFS algorithm.
- Note - you must start with graphs.cpp.
- Grading - 10 HW points. 7 for part a), 2 for part b), 1 for part c).

### HW 9

- Due 4/26
- Late due date: 4/26
- Points: 5, all or nothing
- Pick a kattis problem we have not done in class, solve it, submit the solution online to make sure it is correct, and save your answer in ~/handin/ as filename hw9_PROBLEMNAME.c(pp).
- HW9 checked for: cs202

### HW 8

- Due 4/26
- Late due date: 4/26

- Points: 5, all or nothing
- Checked HW8 for: cs202
- Pick one from signup spreadsheet
  - Once those are taken, go to a linux tutorial common useful commands, or something and pick something else.
- Have a text file called ~/handin/hw8_TOPIC.txt, put your basic example, one or two sentences description, links to further references.

### HW 7a, b, c, d

- Due 3/29 by 11:59pm.
- Late due date: um
- Grading: 4 points each, total of 16.
- For each of - tri.c, speedlimit.c, conundrum.c, everywhere.c
  - make a file *whatever*.cpp that is correct and uses cin/cout/string instead of scanf/printf/char*.  Don't #include stdio.h or stdlib.h (if you do, 0 credit).
  - Note - correct solutions are in /u1/junk/cs202-kinne, or you can use your solution

### HW 6

- Due 3/29 by 11:59pm.
- Late due date: maybe not.
- Counts as 20 HW points - 4 points each
- problems 1-6 in http://cs.indstate.edu/CS201/probs3.pdf
- Note on problem 6 - Exoo says there won't be anything wrong with the input file.  There will be the right number of colons on each line, though a field could be empty.  Thing to do - loop with getline, and inside of that loop with strtok
- Put them in ~/handin/ with the names the file says.  Do exactly as the file says, no extra output.
- Notes…
  - Reading lines - use getline, look in scienceBowl1.c for an example
  - Problem 2 - like finding the smallest integer in an array, except you're reading from a file and using strcmp.  Convert to upper first before strcmp.  (Note - that is a loop). s[i] = toupper(s[i]);

### HW 5

- Due 2/16 by 11:59pm.
- Late due date: 2/26 by 11:59pm
- Counts as 5 HW points
- Grading - Jeff will look at it, and leave some comments.  All or nothing still.
- Copy ~jkinne/public_html/cs202-s2016/linkedList.c into your ~/handin directory.  Also copy linkedList.h and linkedListDriver.c and linkedList.o
- Modify linkedList.c so that the lookup and delete functions are finished and correct.
- Modify linkedListDriver.c to have some different test.

- Compile your solution:
    - gcc linkedList.c linkedListDriver.c -o mine
    - ./mine
- Compile with Jeff's solution:
    - gcc linkedList.o linkedListDriver.c -o jeffs
    - ./jeffs

### HW 4

- Due 2/23 by 11:59pm.
- Late due date: 2/26 by 11:59pm
- Counts as 5 HW points
- https://open.kattis.com/problems/babelfish
- Name the file babelfish.c, and make sure it passes the open.kattis.com website.

### HW 3b

- Due 2/18 by 11:59pm.
- Late due date: 2/26 by 11:59pm
- Counts as 5 HW points
- https://open.kattis.com/problems/oddmanout
- Name the file oddmanout.c, and make sure it passes the open.kattis.com website.

### HW 3

- Due 2/11 by 11:59pm.
- Late due date: 2/15 by 11:59pm
- Counts as 5 HW points
- https://open.kattis.com/problems/everywhere
- Name the file everywhere.c, and make sure it passes the open.kattis.com website.

### HW 2

- Due? Just let me know if you think you have it.
- Late due date: none yet
- Counts as 5 HW points
- Correct working program is called headTail in my class directory.
- Name your program headTail.c in your ~/handin/ directory.
- The program takes 2 command-line arguments. The first is the filename of a text file to open. The second is an integer n. The program should print the first n lines and the last n lines of the text file, with a … in between. Make your output exactly match my program.

### HW 1c

- Due 2/22 by 11:59pm
- Late due date: 2/24 by 8am
- Counts as 5 HW points

- https://open.kattis.com/problems/tri
- Name the file tri.c, and make sure it passes the open.kattis.com website.

### HW 1b

- Due 2/11 by 11:59pm
- Late due date: 2/15 by 8am
- Counts as 5 HW points
- https://open.kattis.com/problems/conundrum
- Name the file conundrum.c, and make sure it passes the open.kattis.com website.

### HW 1

- Due 1/26 by 11:59pm.
- Late due date: 2/3 by 8am
- Counts as 5 HW points
- Correct solution to https://open.kattis.com/problems/speedlimit
- Name your program speedlimit.c in your ~/handin/ directory.
- Make sure to test your program on the kattis website; you'll have to create an account there to do this.  When submitting to the kattis website, make sure to select the option that your program is C and not C++.
- Counts all or nothing.  If the website doesn't accept your answer, it isn't correct.
- A correct working program that you can test different inputs on is speedlimit in my class directory (see the directory at the top of this doc).

### HW 0

- Due 1/18 by 11:59pm.
- Late due date: 1/21 before class starts.
- Counts as 5 HW points
- Login with your cs202xx login.
- On mac use terminal, ssh cs202xx@cs.indstate.edu
- On windows install putty, and login to cs.indstate.edu
- **See Important Links below for logging in and getting putty setup.**
- Run passwd, change your password to something you'll remember.
- Run chfn, and put in your correct full name, leave other information blank.
- mkdir ~/handin
- create a file called ~/handin/hw0_topic.c (so hw0_auto.c will be about the auto keyword), pick a topic from the signup spreadsheet, make ~/handin/hw0.c a small example using that keyword, put a link to where you got it from, put a link to basic information.

## Important Links

See also links in the syllabus

### C and C++

- How to Think Like a Computer Scientist, C++ Version by Allen B. Downey. Required text, that is an introduction to C and C++. Readings will be assigned from the book.
- CS 50: Intro to CS I at Harvard. We will follow this course. Watching video lectures from the course will be assigned.
- Reference on C and C++. Reference on C and C++ language and standard library. The tutorial on the C++ language is very good. Most of the information up until "Classes" applies also to the C language.
- The C Programming Language - the standard reference for C programming. This is good to use as a reference after you have basic understanding of C and programming (about halfway through the semester).
- C Programming Tutorial
- Codepad.org - run basic C and other code online in a web browser. Note that user input doesn't work (cin, scanf, etc.).
- IdeOne.com - another one to run C code online.
- Fresh2fresh C Tutorial - another C reference/tutorial.

### Software and CS Server

- Download Putty. For those using Windows at home, download and install Putty. Then watch the first two videos on this youtube playlist to show you how to use it.
- Getting Started with the CS server - links and videos to help you get started with connecting to the CS server.
- Linux Console Tutorial. A detailed tutorial on how to use the Linux console (command line). Not everything it discusses is available to you (e.g. permissions), but it offers a broad spectrum of what can be done.

### Other Programming

- Scratch Programming - all online, visual, no syntax errors, share with friends!
- MIT Android App Inventor - online, visual, create apps for android.
- Code.org - links to other beginning programming tutorials, many suitable for kids. The organization is pushing teaching programming/CS in K-12 schools.
- Programming at Khan Academy - learn javascript (that is what most interactive websites use)
- Udacity Online Courses - including introduction to CS in Python, intro to Java

# Course Schedule and Notes

## Things you hopefully knew before this course

Note - originally copy/pasted from the spring CS 473/573 Computer Networks course.

### Math

- Arithmetic sum: $1 + 2 + 3 + \ldots + n = n(n+1)/2$
- Exponents:
  - $(b^x)^y = b^{x * y}$
  - $b^x * b^y = b^{x + y}$
  - $b^x / b^y = b^{x - y}$
- Logarithm: $\log_b(a) = x$ means $b^x = a$.
  - For example, $\log_{10}(100) = 2$, $\log_2(16) = 4$. In general, $\log_b(b^x) = x$
  - $\log_b(a^y) = y \log_b(a)$. $\log_2(1000) = \log_2(10^3) = 3 \log_2(10)$
  - $\log_b(a * c) = \log_b(a) + \log_b(c)$
  - $\log_b(a / c) = \log_b(a) - \log_b(c)$
  - Note: we'll almost always use log base 2.
  - Note: log(n) is much smaller than n. Also, $(\log(n))^{100} = o(n^{1/1000})$
- big O means <= up to a constant. So get rid of the constant multiple and it's <=0
  - Rules of thumb: if adding things together, take the one that grows the fastest and remove constant out front. So, $2n^3 + \log(n) + 1000n^2 + 100000$ is $O(n^3)$.
  - Facts: $(\log n)^c$ is $O(n^d)$ for any d, c > 0. $n^d$ is $o(2^{n^\wedge e})$ for any d, e > 0
  - Definition: function f is O(g) if
    - There exists constants c and n0 such that
      for every n > n0, f(n) <= c*g(n)
  - f(n)=10n is O(g(n)), g(n)=n. Want that f(n) <= c*g(n), so 10n <= c*n. That is true if c=10 for all n.
  - $f(n)=n^2$ and g(n)=100n, g(n) is O(f(n))
    - Want that g(n) <= c*f(n) for all >= n0.
    - $100*n <= c*n^2$ for all n >= n0.
    - 100 < c*n for all n >= n0.
    - That is true for c=1 and n=100, or c=100 and n=1.
  - f(n)=10n+log(n), $g(n)=n^2$, is f(n) is O(g(n))?
    - $10n + \log(n) <= c*n^2$ for all n >= n0
    - $10n + \log(n) <= 10n + n = 11n$. Want that $<= c*n^2$.
    - $11n <= c*n^2$ for c=1 and n >= 11.
- Extra garbage you learn in CS 303. So, information for you but I won't put it on a test.
  - little o means <. So, < eventually no matter what constant.
  - big Omega, Ω, means >= up to a constant.
  - little omega, ω, means >.

**table of contents, what we will do today, assignments**

- big Theta, Θ, means = up to a constant.

## Algorithms

- Sorting
    - Heapsort - O(n log(n))
    - Insertion sort (aka slowsort from 201), selection sort, bubble sort - O($n^2$)
- Searching
    - Linear search - O(n)
    - Binary search - O(log(n)), only works on already sorted data

## Data Structures

| | Insert | Delete | Lookup | Notes... |
|---|---|---|---|---|
| Unsorted array | put it at the end of array, O(1) | do lookup, O(n) then swap with last, O(1) | linear search, O(n) | easiest to code. declared with some max size. |
| Sorted array | lookup, shift over everything, O(log(n) + n) = O(n) | lookup, shift over the other way, O(log(n) + n) = O(n) | binary search, O(log(n)) | sorted/unsorted array - if run out of space, need to allocate new space and copy. |
| Linked list (stack, queue) | update a few pointers at beginning of list, O(1) | do lookup, O(n) then update a few pointers (1) | linear search, O(n) | like an unsorted array, but easy to grow bigger |
| Binary tree (balanced) | O(height of the tree), O(log(n)) | lookup, find left-most in right sub-tree to replace, O(height) = O(log(n)) | O(log(n)), the height of the tree is O(log(n)) | but, insert and delete need to do some work to keep it balanced. (Look up AVL tree.) |
| Binary tree (unbalanced) | O(n) worst case if completely unbalanced | O(n) worst case if completely unbalanced | O(n) worst case if completely unbalanced | |
| Heap | O(log(n)) | O(log(n)) | O(log(n)) | |
| Hash table with linked lists for collisions | compute hash, insert into linked list, O(1) avg | lookup, then delete from linked list, | compute hash function, go to that spot in | average<br><br>worst |

| | | | | |
|---|---|---|---|---|
| | and worst | O(size of linked-list) = O(1) avg and O(n) worst | array, lookup in the linked-list there, O(size of linked-list) = O(1) avg and O(n) worst | |

## C Programming

- Any keyword not listed on our "don't know yet" list.
- Basic data types: char, short, int, long, float, double, (unsigned integers)
  - what is the largest/smallest value? how many bytes on cs? (char 1 byte, short 2 bytes, int 4 bytes, long 8 bytes) (float is __ bytes, double is __ bytes)
- Any program that reads a file character by character and does something simple. For example, wc, or count the number of ';' or …
- Types of memory -
  - global variable data and "text" of program - known at compile time
  - stack (local variables) - not known at compile time, grow/shrinks, local variables are stored in a stack (the data structure, so last in first out). OS or C standard library will guarantee some amount stack space, but this amount will take away from the total amount of memory available to everyone else. so stack space should be relatively small. maybe MB.
    - stack overflow - your program tries to use more stack space than guaranteed. nobody checks on this, but eventually you start using memory you shouldn't. runtimer error - segmentation fault.
  - heap (dynamic memory) - not known at compile time, the C standard library keeps track of the heap, you don't get a guarantee on where in the heap you'll get the memory, heap could be shared between all programs
- The way function calls actually work when the compiler converts C into assembly…
  - The stack keeps local variables and also is used to pass information back and forth for function calls.
  - When function is called…
    - push return address (which line in the program gets run after the function finishes) onto the stack.
    - push parameters onto the stack.
    - jump/goto the spot in memory where the function code is.
    - *the function runs, and eventually jump/goto's back to where the function was called.*
    - (after function returns, if it returned a value, pop the return value off the stack.)
  - When function is run/executed…
    - pop parameter values into local variables.
    - run whatever code the function runs.

        - when it’s time to return: pop return address from stack, jump/goto
        - (if value is returned, push the return value onto the stack.)
- pointers, what does BLANK mean?
    - &x means “address of x”
    - *x means “the contents of whatever is stored at address x”
    - x->c means (*x).c
    - x[i] means *(x + i*sizeof(x[0])), aka “value of whatever is at the i th spot in array x”.
        - x could be declared as int *x or int x[100];
    - if declare int *x and want x to be an array, better do
      x = (int *) malloc(sizeof(int) * HOWEVER_MANY_YOU_WANT);
      and later on, do free(x) when not needed anymore.
    - if declare int x[100]; // okay, that’s fine.
    - and note, if you want to return a value from a function, you either use the return statement for that, or have a pointer parameter.
      int f(int *x) { *x = 2; return 3;}
- char arrays, aka strings
    - char *s = “hello”;
      // same as (mostly) char s[6]; strcpy(s, “hello”);
      // “hello\0” is somewhere in memory, and s is set to the address
        - s[0] is ‘h’
        - s[5] is ‘\0’
        - note that in this case “hello” is in a part of memory you are not allowed to change.  can’t do s[0] = ‘z’;
    - char s[7] = “hello”; // I think you can change s[0];
    - Note: regardless of how many characters are allocated, s[10000] is not a compile error.  and program might not even crash, it might just do something that is wrong.
    - strlen(s)
      int i; for(i=0; s[i] != ‘\0’; i++) return i;
    - strcpy(s1, s2);
      int i; for(i=0; s2[i] != ‘\0’; i++) s1[i] = s2[i];   s1[i] = ‘\0’;
    - strcmp(s1, s2)
      …
    - “hello” in a program is really a shortcut for…
      const char temp[6]; temp[0] = ‘h’; … temp[4] = ‘o’; temp[5] = ‘\0’;
      temp is used.
      if you did int x = (int) “hello”; then x is the address of wherever that “hello” array is stored.
- int x[10]; is just like…

  int *x = (int *) malloc(10 * sizeof(int));

  // then use x[1], x[i], etc.

```
// when done
free(x);
```

- difference is where the memory is stored. malloc allocates from the heap. x[10] allocates from the stack if a local variable, and allocates from "data" if global.

## Basic how the computer works stuff

- CPU…
    - Basic instructions are: arithmetic (generally operating on CPU registers), load from memory to register, store from register to memory, if some register is 0 set a flag, if some flag is set do a jump/goto
    - Special register for "next instruction" that holds the memory location to get the next instruction at. Called the program counter (PC).
    - You'll hear people talk about the "load, execute, store" cycle.
- Data on the computer from fastest to slowest:
    - CPU registers (bytes), CPU cache (MB), memory (GB), disk (TB)

## Bits and Bytes and Stuff

- binary: 1100 = $8*1 + 4*1 + 2*0 + 1*0 = 2^3*1 + 2^2*1 + 2^1*0 + 2^0*0$
- decimal: 1100 = $1000*1 + 100*1 + 10*0 + 1*0 = 10^3*1 + 10^2*1 + 10^1*0 + 10^0*0$
- hexadecimal: 1100 = $16^3*1 + 16^2*1 + 16^1*0 + 16^0*0$
- decimal: 4567 = $1000*4 + 100*5 + 10*6 + 1*7 = 10^3*4 + 10^2*5 + 10^1*6 + 10^0*7$
- hexadecimal: D5F2 = $16^3*13 + 16^2*5 + 16^1*15 + 16^0*2$
- why hexadecimal? one hex digit is 4 bits.
- $2^{10} = 1024$. $2^{30} = 2^{10} \times 2^{10} \times 2^{10} = 1024 \times 1024 \times 1024$ = is about $10^9$
- $2^{32} - 1$ is about $2^{32}$ is about $10^9 \times 2^2$ is about 4 billion
- convert from base ten to base b: easiest algorithm is "% b, / b".

## Study Guide for this Course…

### Terms / Notes

- 2's complement
    - number of bits used is required - 8, 16, 32.
    - positive integer: has it's binary value
    - negative integer: take binary of it's absolute value (padded with 0's to however many bits are used to store the number), flip all bits, add 1 (with carrying).
    - adding 2's complement #'s: just add them
    - note: for negative #'s, 2's complement value of x, if you're using b bits, is $2^b - x$
    - note: the first bit is 1 iff the number is negative.

## C++ Programming

- http://www.cplusplus.com/ has good information
- compile with g++
- Most of C remains valid in C++.
- input/output
    - For text, normally use #include<iostream> and use cin for input from stdin, cout for output to stdout, and ifstream and ofstream data types to read/write files.
    - Note - cin/cout and ifstream/ofstream take care of reading the input into the type of variable we're using for us.
- dynamic memory
    - use new and delete, instead of malloc and free
    - You can do malloc/free if you #include<stdlib.h>, but you would use those completely separately from any C++ objects/classes.
    - For string, vector, …, they'll have enough space.
    - But you cannot do
      string x = "hello";
      x[10] = 'h';
    - Instead, do
      string x = "hello";
      x += "blah";
    - Or, use string reserve.
- classes to be familiar with: vector (unsorted array), string (grows on its own!), unordered_map (hash table), map (binary tree), list (linked list), priority_queue (heap), stack and queue (stored as linked lists).
    - string - like a C array, but grows dynamically as needed, can use many times that a char * would be used. and s1 == s2 for string variables s1 and s2 actually does character by character comparison.  and s1 = s2 actually copies the characters from s2 to s1.
- Concepts in C++ that are not in C:
    - classes and object (object-oriented programming)
        - member variables - data inside the class, like in struct
        - class methods - functions inside the class
        - variables and methods have some permission - public, private, protected. public variables/methods - can be used by code that has a instance of the class.  private - can only be used inside of the definition of the class.
        - constructor - function in class with same name as class, no return type, gets run right when the object is created.
    - exception handling
    - polymorphism - functions with same name but different types of parameters.
    - operator overloading - change behavior of operators
    - function parameter passing by reference in addition to passing by value.  C is pass by value only.  nice because can modify value in function without typing *

    everywhere.  dangerous because you have to look up the function prototype to see if a parameter is pass by reference or not.
  - templated functions and types

## Graphs

- Definitions of: vertex/node/vertices (|V|=n), edge/arc/edges (|E|=m), directed/undirected, weighted/unweighted, cycle, tree, connected, degree, planar
- Definitions and basic properties: adjacency matrix, adjacency list
- Running time of doing basic things with graphs.
- Program to read in an adjacency matrix and do something with it.

Unix stuff

- . at beginning of name - hidden file or directory
-

Useful shell commands

- grep "switch" *.c
- pipe is a | and output of the left is the input to the right
- uptime
- top
- ssh-keygen

## What we will do today...

Things will be listed here most recent first.

- For the future
  - Interesting examples can possibly use:
    networking, graphics, databases, *curses*, *big math*, DNA sequencing, some other libraries?
  - 
- Rest of the semester -
  - Algorithms - graph algorithms, recursive examples,
  - Data structures - balanced binary trees, stacks, queues,
  - How the computer works - floating point representation, disk organization (inodes, superblocks, etc.)
  - Using linux - regular expressions, grep, sed, awk, find, etc., pipes and I/O redirection
  - Library functions - directories, stat, etc.,
  - C++ programming -
  - Regular expressions with sed.

**table of contents**, **what we will do today**, **assignments**

    - Note, you've done heaps (priority queues), fast and slow sorting, fast and slow searching, basic stupid data structures (linked list, arrays), hash tables
- 6/17
    - For you to to for more practice, and things we didn't get to but that you really should know…
        - See https://docs.google.com/document/d/1xQIcgyJe1OUkVYqAwLRbVyQJY0TIVonJ5IXVUEhUd-k/edit?usp=sharing for what I think standard assignments for CS 151, 201, 202 are. Pick one you haven't done, do it, then repeat. I am in the middle of developing the list.
        - Do more kattis problems - try to do one per week.
        - Stay tuned for more...
- 5/10
    - Notes from project…
        - If I couldn't find your project.
        - If you plagiarised (use some source in part or in full without citing it) - 0 on the project, and I fill out a form to turn into the academic misconduct folks.
        - Comments - should have comments at top of file for what the program does, how to compile and run it, what works and what doesn't. Should have comments for each function saying what it does, what it returns, what the parameters are. Should have a single line comment (or more if needed) for each loop that has more than a single line body.
    - Notes from second exam…
        - C strings (declared as char *, or char array) cannot be compared with ==.
        - If checking size of a vector, queue, etc., you need to call .size() each time to make you get the latest size.
        - Everyone ended up having their final exam grade higher than the weighted average of all of the exams - both because of the low grades on the second exam, and because people did better on the final.
        - Ask me for a permanent account if you don't have one already.
    - For the summer, things to cover: more C++ object-oriented stuff, make, gprof and/or valgrind, a few more data structures, recursion "try all possibilities" programs, a divide and conquer algorithm, a dynamic programming algorithm, dealing with raw files (maybe "type of file" program, block encryption/decryption), system calls (directories and such), anything that ends up on the list of things for 151/201/202 after that is put together this summer.
    - For the summer, things for you to review: basic C and C++ programming (keep doing contest problems regularly), the math you're supposed to know, the basic data structures (basic descriptions of operations, running times of operations)
- 4/28
    - Jeff to do
        - something about things we were supposed to do that we didn't.
        - HW 10 grade

    - Note - final exam will be <= second exam in terms of difficulty, and you'll have 2 hours instead of 90 minutes.  Also, will have more of the problems that are very similar to something we've done.
    - Come, you get practice on paper, for 50 minutes.  Someone (Aaron Cox) will be hear to give you the practice problems.  You'll turn them into the lab.  And if you want to ask something about anything, you can ask Aaron.
    - Study for the final, do the project.  Our final is Thursday.  So come by, ask by email, etc.
    - I'll be out of town Thursday through Monday.
- 4/26
    - HW 10 - anyone done? So, do that.
    - Other time to meet?   Noon-1pm next week nobody has finals.  Tuesday and Wednesday noon-1pm, I'll be here.
    - AVL tree.  Start with a plain binary tree and then do https://en.wikipedia.org/wiki/AVL_tree
    - Your projects?
    - Second exam …
        - General test-taking strategies - check your answer right after you did it, do your work on the paper neatly (increases chance you get the right answer, and increases chance I give you half credit, maybe takes less time)
        - Note: hash tables are fastest, why would you use a balanced binary tree? Uses less memory, and you can find the next if that's something you would need.
        - Grades on paper, ranged from close to 0 to close to 10.  If I gave half credit for more things, then low grade may have been more like 5 or 6.  A 2/10 with the way I graded means something like 20% of my questions you can solve without mistakes; in real life, maybe you would get something like 70% of your programs without mistakes.
        - Grades - after I put in grades for computer, will update everything, give an estimate.  For people repeating, you're not going to get lower than a C as long you try for the next week.  For people not repeating, convince me not to give you lower than a C.
        - 202 is a prereq for everything else.  I'm supposed to be certifying that you are ready, and in other courses they won't have to teach you how to program and use data structures.  Next year both 201 and 202 are 4 credits.
- 4/21
    - Second exam. Exam over at 12:25pm
- 4/19
    - Attendance -
    - Anything of anyone's need to be graded?
    - Second exam - this Thursday the 21st
        - Everything we have done.  This includes...

        - First exam style questions.  Any quiz.  Any HW assignment.  Anything in these notes.
        - New topics/things on this exam
            - Graphs - basic properties, BFS, DFS, simple programs dealing with adjacency matrix or adjacency list.  Do a program to take an adjacency matrix and compute the degree of each vertex, something else.
            - C++ programming - create a class, use a class already given, cin, cout, ifstream, ofstream, stack, queue, vector, string, parameter passing by reference/value, new/delete,
            - If I come up with anything I want to ask about, I'll let you know.
        - Definitely not on the test - curses, sqlite
        - Exam grade is max of (don't drop anything, drop exams 1 and 2, drop exam 1).
        - Same basic outline with paper and computer parts.
    - Sqlite example
    - RPN calculator
- 4/14
    - Heh, spend at least 20 hours per week programming.
    - 
    - Finish load average program, using curses.
    - Another example maybe using SQLITE and statistics coming from last
    - Look at examples of using C++ stack, queue, vector.  And if comfortable with those, map, unordered_map, priority_queue.  Copy/paste the example into your account, compile/run/debug/modify/debug/run/etc.
    - And next after that - balanced binary tree (AVL trees).
- 4/12
    - Second quiz on graphs. max(q4, q5) counts.
    - Note rest of the semester - more definitions, math, algorithms.  For those - quizzes.  And more programming stuff.  For the programming stuff - I do "projects".   Some of these programs in /u1/junk/cs202-jkinne/jkinne-project/
    - Note - lookup something like "linux ssh key login".
    - For this time, my project was load for all CS machines.  Things to fix…
        - Error message with ssh
        - Put in curses and auto update
        - Include other CS machines.  Possibly auto get list from somewhere.
        - Other information
- 4/7
    - More HW9, HW8's?
        - HW8 awk, fix the script or give a different example?
    - New HW10 definition.
    - Note on BFS, DFS

table of contents, what we will do today, assignments

        - BFS is shortest path on unweighted graphs. Is shortest path on weighted graphs if keep list of vertices need to visit in a priority queue (aka heap).
        - DFS used in path finding, AI. (Actually, depth-limited DFS, and other variations).
    - Issues with alias, locate, find from last time?
    - Project…
        - Pick a topic, learn it, work on a program. Start as simple as possible, make sure it is working.
        - Come talk to me before Tuesday about what you want to do. Note - you are forced to find out where my office is. And we can work on it for longer than 5 minutes. Times - MWF 8:30-3:30, TR not 9:30-12:30.
        - Some requirements - it has to use stuff. I'll have to give you a grade.
        - Topics -
            - Graphics - gtk or openGL or something else. Note that you'll need to run it while physically at these machines. Or, you have linux computer, or you have cygwin on Windows, or you have an X server running on Mac.
                - Gtk: see https://developer.gnome.org/gtk3/stable/gtk-getting-started.html and also gtk_example.c in the usual directory of mine.
            - Database - sqlite (database inside of a file, no need for users or passwords). Maybe find some sqlite database you can start with. Later you could create your own. Some program to look up (and modify, insert) information.
                - See https://www.sqlite.org/quickstart.html and also sqlite3_example.c in the usual directory of mine.
                - http://zetcode.com/db/sqlite/
            - GMP - arbitrary precision math. Computing millions of digits of pi, e, sqrt(2). Doing huge integer stuff (prime testing, RSA, …).
                - See https://gmplib.org/manual/ and also gmp_example.c in my usual directory.
            - Networking - sockets (html downloader, ftp, smtp)
                - See for example http://www.linuxhowtos.org/C_C++/socket.htm and sockets_example.c in my usual directory. There are other options as well.
            - Work on cards program. Finish it, make lots of grames (blackjack, poker, hearts, solitaire).
            - Work on science bowl questions program. A lot of string processing.
            - Curses game - window stuff in putty. Looks like old DOS games.

            - See http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/ and also mazeGame2.c in the usual directory.
        - Spelling quiz - my 11 year old needs help with spelling.
        - Applications in WIndows with MS Visual Studio
            - Yep, find a tutorial and do it, as a start.  Note that MS VS Express is free.
        - Other ideas - check with me.

- 4/5
    - HW8, HW9, HW10.
    - Questions
        - Alias with color? Or what?
        - Find all files in my directory that start with hw8
        - And locate?
        - Difference between find and locate.  Locate uses the database.
- 3/31
    - HW10
        - Graphs - take graphs.cpp, and
            - add DFS.  Instead of queue, use stack.
            - Add a method that checks if the graph is connected (run BFS, and then check for INT_MAX distances).
            - Add a method that checks if the graph has a cycle. Use the BFS algorithm.
            - Note - you must start with graphs.cpp.
    - Note - we're going to be moving faster now.  Spend all your time on this class.
    - Quiz on graph definitions and basic concepts - see above.
    - Paper exams returned.  / is -.5.  Some grading notes listed above where the exam is in the syllabus.  Total on the back is # of points off, score in BB should be 10 minus that.  Make sure I didn't make any mistakes.
    - Jeff check hw7 **tri** and hw8, and exams problems again.  Done.
    - Graphs - BFS, DFS.
        - Example graph - http://lh5.ggpht.com/-KKS-GOBQJ0o/T_11xTcy6CI/AAAAAAAAAb0/yHB5Au4YZ5s/BFSGraph_thumb2.png?imgmax=800
- 3/29
    - Check your HW6, HW7, exam1 computer grades.  If something doesn't seem right, let me know.
    - BFS - https://en.wikipedia.org/wiki/Breadth-first_search
    - DFS - https://en.wikipedia.org/wiki/Depth-first_search
- 3/24
    - Look up that stuff about using linux, read about C++ classes.
    - Practice some programming - pick a kattis problem we haven't done, solve it (without searching for the answer online), we can count that as HW9.

**table of contents**, **what we will do today**, **assignments**

    - Finish unordered_map example in C++.
    - Back to the cards example - war, go fish.
    - And big math or curses, or networking, or databases, or DNA sequencing.
- 3/22
    - HW 8 - pick one of the commands, and give us a simple useful example, a pointer to information (tutorial, cheat sheet).
    - 
- 3/22
    - Builtin C++ data types (vector, etc.). What is a stack and a queue? Things we still need to talk about.
    - Read again the stuff in the tutorial about classes.
- 3/10
    - You to do: read http://www.cplusplus.com/doc/tutorial/ starting from Classes. Try out the examples - type them in, compile, run. If errors, debug.
        - Note: if you don't, what are you here for?
    - What about malloc/free in C++
        - Generally,
- 3/8
    - You went over the correct answers for the test last time. I'll get them back to you next time.
    - Today: C++...
        - Read through http://www.cplusplus.com/doc/tutorial/
        - Lots of example programs
        - See some notes above...
- 3/3
    - HW7 - something with our hash table, still in C
    - switching to C++ - I/O in C++, dealing with strings, and using builtin C++ data structures, templated functions/classes in C++ (the coolest, nicest thing about C++).
    - HW8 - laundry list of basic C++ programs.
    - HW9 - contest problems in C++.
- 3/1
    - Exam 1 - see information for "Exam the first" in the table of contents at the top.
        - test over at 12:22pm.
    - Note: HW6 you can still do.
- 2/25
    - study list - 2's complement, hex to binary to decimal, pointers, arrays, big-O stuff for all the data structures.
        - All of those will be on the test.
    - linkedList, babelfish - anyone else done? linkedList S, babelfhish T?, oddmanout T?
    - linkedList and babelfish due by Saturday midnight.
    - Other HW's not past due - HW6, HW3b - oddmanout. due by Saturday midnight.

        - HW1c, tri.c - everyone got it now.
        - **Note: moved solutions to contest problems to /u1/junk/cs202-kinne so they are not publicly available (so other courses can use the problems without worrying about people searching for the answers).  Don't share the solutions publicly!!!**
        - Exam next time.  Questions ?
        - hash table example, other improvements - deal with punctuation better, keep track of longest bucket and total # of items.
        - binary search tree.  useful when need to have items in order (e.g., routinely need to find the next or previous item).
        - heap - useful when need to routinely have the least or largest element, often used in algorithms (e.g., shortest path).
- 2/23
    - babelfish - tell me if you have submitted to the website and it is being judged as correct.
    - linkedList - S and K done.
    - interim grades - a few A's, some C's, D's, F's.  The interim grade is what you would get if the semester ended now.  Hopefully you can improve on that.  Some people are trying, others are not.
    - extra programming practice - check http://cs.indstate.edu/CS201/
      Some of those assignments already have correct solutions provided by Dr. Exoo. Look in /u1/junk/cs201/ for what the 201 class is up.
    - HW6 - CS 201 problems 1-5 due Feb 26.  There may not be late credit. Jeff - ask Exoo not to put the solutions out.
      http://cs.indstate.edu/CS201/probs3.pdf
    - use the hash table to do word frequency counts in shakespeare.  output top 10 most frequent. that would need having an extra field in ll_node
        - Yeah, woo hoo.  10 is a good ratio for hash table buckets versus # elements.
        - Things to do for HW - print out statistics (longest linked list bucket, average length of linked list, # occupied buckets), fix the checking of words to get rid of punctuation, better hash function?
    - For you to do for fun -
        - do the same thing as hashTableDriver.c, except use an unsorted array for the data structure.  It should be a lot slower.
    - what else?  uh ….
- 2/18
    - exam?
        - yes - when - March 1.
        - topics - everything we've covered.  All syntax in C.
        - half on paper, half on computer.
        - Question types

                - C program, and what is the output, or values of variables, or how many times a loop ran, or how much memory is used, or something else.
                - Complete a C program - I give you most, you finish it.
                - C program from scratch.
                - Use one of the data structure files we've created to do something.
                - Math - logs, powers, sums, hex/bin/decimal, big O
                - Data types - binary, 1's complement, 2's complement
                - Data structures - the ones above.
    - for your fun - modify the qsort_example.c function to sort using insertion sort - put the code for insertion sort (or selection, or bubble, or $n^2$ algorithm). Try it with howMany=1000, 10000, 100000. Something to share - what value of howMany does "bad sort" still finish in less than 10 seconds? And what value of howMany with qsort. Use time before your command.
- 2/16
    - Jeff - put in grades for people who have headTail (only three so far)
    - Note - quiz3 returned next time.
    - homeworks - conundrum, everywhere past late due date. see model solutions for those. babelfish - use qsort and bsearch. linked list hw - questions?
    - New kattis problems - hw 1c and hw3b, both due in 1 week. hw1c is of similar difficulty to the hw1 problems, hw3b is of similar difficulty to the hw3 problem.
    - hash table - finish, and note modifications to linkedList.h/c
    - data structures in C++? second half of the class.
- 2/11
    - homeworks - if you already have full credit, I'll send you my solution if you ask. Don't share my solution. View my solutions as "reasonable" but maybe not perfect.
    - quiz 3 - note that hash tables have O(1) time operations on average if things work out well. worst-case for hash tables is O(n).
    - keep working on hw's already assigned.
    - also, you will need to finish linkedList.h, use linkedListDriver.c as a test. I'll make up a HW assignment by next time.
    - and the sciencebowl stuff...
- 2/9
    - quiz 3 - similar to quiz 2, but …
    - questions on hw's? Jeff - run the scripts to grade the hw's.
        - hw2 - no due date, let me know if your program is exactly the same as mine.
        - hw1b, hw3 - due tonight. Anyone think you have it?
        - hw4 - due Thursday maybe. Anyone think they have it?
    - scienceBowl - Jeff has a reason to require a hash table, has completed compiled code, and partially completed source code.
- 2/4

    - fixed sciBowl1.c program.  problem was in insert - need to allocate storage for string; that also means need to free in delete.
    - start working on one of the other data structures, hw5 will be for you to finish it...
    - Note - you should know the multiplication facts.  If you need practice, then practice them.  See the internet for practice apps, or http://cs.indstate.edu/~jkinne/kids/arithQuestions.php
    - hw1 past due now.  my answer is speedlimit.c in the usual directory.
        - Note: you should be concerned if you could not get this program.  You should panic, and you should spend more time on programming.  Look at my program, and then try to recreate it from memory so it will pass the kattis website.  And get hw1b correct.  Now is the time to start spending all your free time programming.
    - hw1b, hw3, hw4 - see hw section.
    - hw2 - make sure to print a newline at the end.  check my program.
        - note: run your program and > to save results, run my program and > to save results, use diff to compare.
- 2/2
    - hw2 - not yet graded?
    - Quiz2 over basic data structures.
    - Also you do some big-O practice.  Search for “big O examples”.  Also
        - If you want a book about this kind of stuff, get a “discrete math” book or a “algorithms” book.  Discrete Math and Its Applications by Rosen.  Introduction to Algorithms by CLRS (Cormen, L, Rivest, …)
        - If anyone finds a useful source, let me know, I’ll share the link.
- 1/28
    - hw2 - Jeff puts in a grade this afternoon, if you get a 0 and think you’re right let me know.
    - For you to read for your indoctrination - assembly language, Von Neumann machine, CPU register.
    - Maybe a little with the sciBowlQuestions.txt but also…
    - Review of the basic data structures - see above, and look up on wiki.
        - Quiz on 2/2 over basic data structures - not using them, but one line description of the operations and running time.
- 1/26
    - Take a look at strtok again, and ponder…
    - Take a look at the register keyword, I’m allowed to use it on the next quiz.
    - All of the C keywords are allowed.  All of the operators?
    - hw1 and hw2.  questions?
    - finding segmentation faults - lots of print statements, or gdb
        - compile with -g flag
        - run gdb with argument that is name of executable
        - type “run” in gdb to start the program and include command-line parameters (e.g., “run argument1 argument2”)

        - when it crashes, type backtrace to see the list of functions that have been called.  if you type “frame 6” it loads up the function that was listed as #6 in the backtrace output, and then can do “print x” if x is a variable in that function.
        - look at stack trace and line numbers.
        - search online for commands to look at values of variables, set breakpoints, etc.
        - Note - IDEs like MS visual studio or Eclipse will provide a graphical debugger that is easier to use.  gdb does all the same things, you just have to type commands instead of point and clicking.
    - 
- 1/21
    - hw0, whoever didn’t go last time.
    - quiz1 - you’ll get it back.
        - note: I can’t give half credit if you don’t show your work.
    - getting started on science bowl questions.
        - will start with program that reads the file, computes how many lines, characters.  You should do that before class.
        - first step is getting out all the different categories.  say, output the category of the first 30 questions.
- 1/19
    - hw0 due 1/18 by midnight, at the beginning we’ll look at those.
        - If you didn’t have anything yet, it’s late, get it for next time.
        - Also, see “goto considered harmful” for fun.
        - those who had it done today, full credit.  those who do it next time, 80%, and that’s your last chance.
    - quiz 1
    - Start doing data structures.
    - First running example - science question generator.
- 1/14
    - Quiz 0 - see notes above about what is on the quiz.
    - We did already...;
        - switch/case/break/default, ?: (aka the inline if expression),
    - Keywords, operators/tokens in C - any of them you don’t know how to use?
        - keywords: auto, const, continue, enum, extern, goto, register, signed, sizeof, static, typedef, union, volatile,
        - operators: type-casting
- 1/12
    - People want to take picture of board for later reference?
        - Note - it has been shown that for most people taking notes helps you remember things better ...
    - Syllabus, blah blah

        - Things you should know… - math stuff above, binary/decimal, linear and binary search
        - Things you should know that you didn't - hex, base 16.  Convert between binary and hex, between hex and decimal.
        - Textbook - nope, sorry.  For C/C++, use the sources listed.  For algorithms/data structures, look at wikipedia first.
    - Get your cs202xx login from BB and login to the computer
    - Attendance in BB quiz
    - 
    - Keywords, operators/tokens in C - any of them you don't know how to use?
        - keywords: auto, const, continue, switch/case/default, enum, extern, goto, register, signed, sizeof, static, typedef, union, volatile,
        - operators: type-casting, ?:,
    - Commonly used libraries in C - any of them you don't know how to use?
        - you have used: printf, scanf, fopen, read, write, fprintf, fscanf, strlen, strcmp, isupper, islower, isalpha, isdigit, isalnum, tolower, toupper, abs, min, max, sqrt, log, sin, cos
        - you maybe have not used: getc, putc, ...
    - HW 0 assigned
    - Quiz 0 next time
        - we'll do a quiz, I'll tell you what it's over, then we'll do another one, and the score that counts is the max of two.
    -

## Email Log

If I remember, I'll copy/paste emails to the class here so you can refer back to them in case you accidentally deleted one.

- 1/21
    - I put in two programming assignments, hw1 and hw2, into the google doc.  I have them as being due Tuesday by midnight, so get a start on them before Tuesday so you can ask if you have questions.  If those who have spent some time on it need another day, we can talk about that on Tuesday.  They are not really related to the science bowl stuff, but just practice programming (which you should want to do).
- 1/11
    - Cheers.  You're getting this because you are enrolled in CS 202.  I have put items in the blackboard gradebook with your cs202xx login and password.  You should

check the grades in your BB for this course and try those out to connect to the CS server, so you'll be able to use the lab computers tomorrow during class.

- Syllabus and everything else isn't ready yet, we'll do that tomorrow.  See you then.