Course Syllabus for CS 131 – Introduction to Computer Science I – Fall 2010

Basic Course Information:

Instructor:	Sharon Tuttle		
Lecture times & location:	Tuesday, Thursday	2:00 - 3:20 pm	GH 218
Lab times & location:	Section 11: Friday	8:00 - 9:50 am	BSS 313
	Section 12: Friday	10:00 - 11:50 am	BSS 313
Instructor's office:	BSS 322		
Instructor's e-mail:	st10@humboldt.edu OR sharon.tuttle@humboldt.edu OR smtuttle@humboldt.edu		
Instructor's office phone:	(707) 826-3381		
Instructor's office hours:	Monday, Wednesday	3:15 - 4:45 pm	
	Tuesday, Thursday	9:30 - 10:30 am	
	or by appointment		
Course public web page:	follow link from: http://users.humboldt.edu/smtuttle/ OR follow link from course Moodle site		

Course Description:

[from the HSU catalog] Concepts; historical background; computer systems; algorithmic processes; control structures; scalar data structures and arrays; structured programming in C++.

This course is an introduction to Computer Science that happens to use the Racket and C++ programming languages. There will be a strong emphasis on problem-solving and on good program design, and object-oriented programming will be introduced.

Students are expected to come into this course already-comfortable with basic microcomputer computer use. **No prior programming knowledge is assumed or required.**

Course Objectives:

After successfully completing this course, students should be able to: *

- Design, implement, test, debug, and use functions and classes, following a standard design recipe.

* Adapted from the ACM Computer Science Curriculum 2001, available from link at: http://www.acm.org/education/curricula-recommendations

- Analyze and explain the behavior of simple programs involving the fundamental programming structures of sequence, condition, iteration, and procedure, and involving simple classes.
- Modify and expand short programs that use standard conditional and iterative control structures, auxiliary functions, and classes.
- Design, implement, test, and debug functions and classes that use each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions.
- Choose appropriate conditional and iteration constructs for a given programming task.
- Apply the techniques of structured (functional) decomposition to break a program into smaller pieces (or, better yet, to design it from smaller components to begin with!)
- Describe the mechanics of parameter passing.
- Discuss the importance of algorithms in the problem-solving process.
- Identify the necessary properties of good algorithms.
- Create algorithms for solving simple problems.
- Use pseudocode or a programming language to design, implement, test, and debug algorithms for solving simple problems.
- Describe strategies that are useful in debugging.
- Discuss the representation and use of primitive data types and built-in data structures.
- Describe common applications for primitive types, arrays, and strings.
- Describe the concept of recursion and give examples of its use.
- Identify the base case(s) and the general case(s) of a recursively-defi ned problem or function.

Course Prerequisites:

Math 115 or MPT3 15 or Math code 50 or instructor consent.

Required Course Text, Materials, etc.:

- Turning Point RF Response Clicker, available at the campus bookstore, or a Responseware license, available on-line (address to be provided).
- "How to Design Programs", Second Edition, by Felleisen, Findler, Flatt, and Krishnamurthi, currently only available on-line, at: http://www.ccs.neu.edu/home/matthias/HtDP2e/
- Links to additional on-line required readings may be provided.

Recommended Course Text:

- "Problem Solving with C++: the Object of Programming", Seventh edition, by Savitch, published by Addison-Wesley.
  - (Note, however, that this book is being recommended just as a C++ **reference** for students who

would like such a reference – no required reading assignments will be made from this text. Also note that you will **not** need the CD that comes with this text.)

Course Software:

For the first part of the semester, you are expected to use subsets of the **Racket** programming language using the **DrRacket** programming environment. This software is available from:

http://www.racket-lang.org/

...and has versions for Windows, Linux, and Mac OS X; it should also be available in the following on-campus labs:

BSS 313, BSS 317, FOR 204A. LIBR 121, and LIBR 122

For the remainder of the semester, you are expected to use the **GNU C++ compiler** installed on nrs-labs.humboldt.edu. This, too, is free software, although we initially will be using it in conjunction with some customized C++ tools on nrs-labs. But, this GNU C++ compiler is, for example, included in Dev-C++, a free C++ environment for Windows also available in the BSS 313, BSS 317, FOR 204A, LIBR 121, and LIBR 122 labs and available for download (for Windows only) from:

http://sourceforge.net/projects/dev-cpp/

(Mac OS X users can download a version of Xcode from the Apple Developers Connection (use the free option at:

http://developer.apple.com/programs/register/

), which includes this same GNU C++ compiler, if their computer doesn't already have the GNU C++ compiler. Mac OS X, Linux, and Windows users can also download the C++ version of Eclipse from:

http://www.eclipse.org

...which also uses this same GNU C++ compiler.)

So, if you were to develop preliminary versions of your C++ coding in Dev-C++ or on a Mac or under Linux, such code ought to run on nrs-labs as well -- but it is **your responsibility** to verify that this is the case for your code before submitting it.

Throughout the semester, you will be making some use of the UNIX operating system (since that's what nrs-labs.humboldt.edu uses). Instruction on how to do so will be provided. Note that you can access nrs-labs.humboldt.edu by using **ssh** (secure shell) and **sftp** (secure ftp); ssh may be downloaded for free from:

https://experts.humboldt.edu/ditss/download/ or
http://www.humboldt.edu/its/software (use this one for graphical versions for Mac OS X)

...although in Fall 2008 some students had better luck downloading it from:

http://www.colorado.edu/its/docs/authenticate/printouts/win_ssh.html

...which also includes a nice illustrated tutorial for this Windows implementation of ssh and sftp.

Clickers:

We will be using Turning Technologies student response clickers (or, for those who prefer, Responseware on one's cell phone or laptop) in lecture and sometimes in lab. There is significant literature indicating that using clickers may increase student engagement and success in learning.

Students purchase this clicker or buy a Responseware license; purchased clickers can be returned at the end of the semester for a partial refund of the purchase price. Students with clickers register them once, at the beginning of the semester, by entering the large number (consisting of 6 characters/digits) on the

back of the clicker at a special address that I will provide, and then bring them to every class meeting. Students using Responseware purchase the license, and then sign into Responseware at the beginning of each class meeting, using a special code that I will project at the beginning of each class meeting.

These clickers will be used for in-class questions, which will be interspersed during the lecture and lab sessions (although probably with more questions during lecture sessions, since lab sessions may also include lab exercises). The response system will record the overall class response percentages as well as keep track of individual answers. Students will receive **2 points** for a correct answer, **1 point** for an incorrect answer, and **0 points** for no answer, but with a maximum semester clicker-questions grade of **120**. Thus you will be rewarded for regular attendance and participation. If you miss a class session, you miss that day's clicker questions and cannot make them up. However, there will be at least **65** questions asked over the course of the semester, so there is opportunity for extra credit (up to a maximum clicker grade of **120**) (or to make up for a day that you are out due to illness, although note that you are still responsible for finding out what you missed on such days).

If you forget your clicker for a class meeting, then **up to 5 times** you may still receive some clicker credit, **minus a 2-point penalty**, by e-mailing me your clicker answers for that day, **by midnight on that day**, using the Subject: **CS 131 Clicker Answers for <date>**. Later e-mails, or e-mails without the proper Subject: line, will not be accepted for credit.

The idea is that the clicker questions will help you to see if you are starting to understand concepts being discussed; sometimes they will also provide review of concepts discussed previously. Clicker questions are typically quite different from exam questions (since clicker questions are typically multiple-choice questions, while exam questions will rarely be multiple-choice). They still enable you to get some immediate feedback regarding whether you are grasping course concepts, whether you need to pay more attention to course discussions and/or readings, etc. They may even help me to know what concepts might need more explanation in-class.

I hope to run tests of the system during Thursday's lecture and Friday's labs during the first week of the semester, and hope to begin asking questions that "count" during the second week of the semester. Therefore, you must purchase your clicker and register it (or purchase a Responseware license) as soon as possible. If there is an issue with this (for example, if the bookstore runs out of clickers), contact me immediately.

Finally, please note that use of another CS 131 student's clicker, or having someone else use your clicker in CS 131 class -- that is, pretending that a student is in class who actually is not -- is considered to be cheating, with the same policies applying as would be the case if you turned in someone else's work as your own or permitted someone else to copy your work. Please ASK ME if you are not sure what I mean by this.

Grading Breakdown:

If you are a Computer Science (CS) major or a Computer Information Systems (CIS) major, **note** that you must earn **at least a C** in CS 131 for this course to count towards your major.

Your semester grade will be determined by the percentage of points that you earn, **subject to some minimum requirements**. Here are the grade percentages, followed by those minimum requirements:

Homeworks assignments:		25.0%	
Lab exercises:		12.5%	
Clicker questions:		12.5%	
Exams:	**Exam #1:**	15.0%	
	Exam #2:	15.0%	
	Final Exam:	20.0%	Thursday, December 16, 3:00 - 4:50 pm, GH 218

Grade Requirements:

1. To earn a grade of **C or better** in this course, the following three requirements must **all** be met:
   - your overall semester average must **equal or exceed 72.5%** - this is to show a reasonable level of overall mastery of the course material.
   - the **average** of your Exam #1, Exam #2, and Final Exam grades must **equal or exceed 60%** - this is to show that you understand at least a minimal reasonable level of the most important course concepts.
   - the **average** of your Homework assignments must **equal or exceed 60%** - because this is a programming course, but programming acumen is not tested as effectively on exams, this is to show at least a minimal level of programming competence and experience in addition to course concept mastery. Also, past experience has shown that students who do not put a solid effort into the course programming assignments do not do well on the course exams.
2. If **all three** requirements above are **not** met, then your semester grade will be **either C-** or the letter grade computed according to the mapping given below, **whichever is lower**.

   (That is, if a student had an overall semester average of 74% but a Homeworks average of 55%, that student would receive a **C-** for his/her semester grade; if a student had a Homeworks average of 61% and an Exams average of 71%, but an overall semester average of 65%. then that student would receive a **D** for his/her semester grade. You are expected to ASK ME if this aspect of the grading policy is not clear to you.)
3. Including the three requirements noted above, your semester grade will be computed according to the mapping given below:

Overall Percentage (based on the given weights)	**Exams Average**	**Homeworks Average**	**Letter Grade**
>= 93	>= 60	>= 60	A
>= 90 and < 93	>= 60	>= 60	A-
>= 87 and < 90	>= 60	>= 60	B+
>= 83 and < 87	>= 60	>= 60	B
>= 80 and < 83	>= 60	>= 60	B-
>= 77 and < 80	>= 60	>= 60	C+
>= 73 and < 77	>= 60	>= 60	C
>= 73	< 60	any	C-
>= 73	any	< 60	C-
>= 70 and < 73	any	any	C-
>= 67 and < 70	any	any	D+
>= 60 and < 67	any	any	D
< 60	any	any	F

Final Exam:

Again, the Final Exam for this course is scheduled for THURSDAY, December 16th, 3:00 – 4:50 pm, in GH 218 (unless I announce otherwise). Note this time and date BEFORE making your end-of-semester travel plans.

Students with Disabilities:

Persons who wish to request disability-related accommodations should contact the **Student Disability Resource Center** in the basement of the Library, Library 055, **826-4678 (voice)** or **826-5392 (TDD)**. Some accommodations may take up to several weeks to arrange. You can reach the Student Disability Resource Center's web site at:

http://www.humboldt.edu/disability/

Please note that some accommodations may take up to several weeks to arrange; also, if you are eligible for such accommodations, please contact me as soon as possible to discuss them.

Add/Drop Policy:

Students are responsible for knowing the University policy, procedures, and schedule for dropping or adding classes. You can find these on the web at:

http://www.humboldt.edu/registrar/students/regulations/schedadjust.html

You can find the University policies for repeating classes at:

http://www.humboldt.edu/registrar/students/regulations/repeat.html

Note that the CSU (and thus HSU) policies on withdrawing from and repeating courses changed as of Fall 2009:

- Students may withdraw from no more than 18 semester-units after the first four weeks of instruction; that is, students may withdraw from no more than 18 semester-units between census and the final 20% of instruction, and only then with a serious and compelling reason.
- Students may repeat courses only if they earned grades lower than a C.
- Students may repeat up to 16 semester-units with grade forgiveness.
- Students may repeat up to an additional 12 semester-units with grades averaged.

Be careful – as of Fall 2009, HSU is being much more strict about what constitutes a "serious and compelling reason".

The census date for Fall 2010 (before which you can drop without a W, and without it counting toward your 18 semester-units drop limit) is: **5:00 pm on Monday, September 20th**.

The last date for Fall 2010 to drop with a W on your transcript, with a serious and compelling reason, and subject to the 18 semester-unit drop limit, is: **Friday, November 12th**.

If you do drop the course, note that it is **your responsibility** to complete and submit the appropriate paperwork. As noted in the University policies for dropping courses, "As a matter of university policy, **the instructor in the course may not drop on behalf of the student**."

Incompletes:

Incompletes are rarely given and only in the case of a true emergency. They certainly are not appropriate for students who find they have fallen behind on assignments, missed a test, or taken on too

much academic, work, or family responsibilities. For these situations, dropping the course would be appropriate (if that is still possible according to the University policies for dropping courses).

Time Expectations:

Remember the general rule of thumb for college-level courses --- to be successful in a course, you should plan to spend at least 3 hours outside of class for each 1 hour of college course credit. That implies an estimate of at least 12 hours a week spent outside of class for this 4-credit course.

However, you should be warned that:

- Programming courses can be notorious time eaters – occasionally, a problem with your program will take large amounts of time to locate and fix. Starting early enough so that you have time to ask me questions when you run into problems can help with this. Why spend 4 hours struggling with a frustrating roadblock the night before the assignment is due, when you can spend 10 minutes composing an e-mail early in the week, work on other problems while waiting for the answer, and then get a reply that makes everything clearer as soon as you read it?
- You can only learn programming by practicing it, and it takes some much longer than others to master it. Practicing programming as much as possible helps. (This means playing around with in-class examples, experimenting to see if something you are curious about really works like you think, and so on.)
- The course will intensify as the semester progresses – as you are able to do more, you will be expected to do more. If you ask me as soon as you realize that some concept is not clear to you, that can help keep you from falling behind.
- Homework deadlines will not be extended because you waited too late to start or because you did not allocate enough time before the deadline to work on it; likewise, they will not be extended because of hardware or network failures. You need to keep backups of your files at all times, and need to plan your schedule to be able to work on on-campus computers as necessary.
- If you have not completed an assignment by the deadline, your best choice is to submit whatever you have managed to do by then, as partial credit is your friend, to carefully study the posted example solution as soon as it is available, to ask me about anything there that is still unclear, and to get a good early start on the next homework.

Academic Honesty:

Students are responsible for knowing policy regarding academic honesty:

http://www.humboldt.edu/studentrights/academic_honesty.php

Observe that among the actions that are unacceptable are submitting another's program, code, or file as your own and failing to quote material taken from another person's written work. (Note that copying another student's comments is also unacceptable.)

All course work is to be the work of each student, **individually**, **unless** it is **explicitly** stated otherwise at the beginning of that course work's description. Except for explicit exceptions, this is **not** a group or team programming course. When group work is explicitly permitted, the names of all students involved must be included on the work submitted. (For example, when you use **pair programming** in lab, the lab exercise will specify that, and the files you turn in will include both of the names of the students who worked as a pair.)

(When an assignment does specify that it is acceptable to work in pairs or groups, make sure that you don't get into the situation where you are merely watching someone else learn.)

For homework assignments (that are not specified as pair-programming), students may discuss general

approaches **as long as no one involved in the discussion is writing anything down or typing anything during such discussions**. Students may also help one another in determining causes of program bugs, or in determining the meaning of compiler error messages. However, students may not work together to complete homeworks, one student should not instruct another in how to write the code for a homework, and **any type of copying or modifying of another person's computer files, OR of providing computer files to another, related to homeworks is definitely over the line, and never justified**. This applies to copying of documentation and comments as well as to copying of program code.

Note that it is **your** responsibility to ensure that course assignment files are read-protected. If you are careless about this, and someone else copies your work, you will share the penalty. (In particular, be very careful about leaving work on shared network drives in campus labs, or in UNIX/LINUX directories that are not read-protected.)

Learning takes hard work; when students turn in others' work as their own, it is a slap in the face to those seriously interested in learning. Not turning in an assignment results in no credit for that assignment, of course, but that is an honest grade. Work that violates the course honesty policy deserves a lower grade than that, and therefore the course policy is that work violating this policy will receive **negative** credit. A person providing a file for copying receives the same **negative** credit as the copier. Repeat offenses will be handled according to University policies.

Asking Questions/Getting Help:

- You are encouraged to ask me questions in class, in office hours, and by e-mail. The most successful students are those who are not afraid to ask questions early and often (I will gently let you know if you are overdoing it), who do the assigned reading, who attend lecture and lab regularly, who start assignments **promptly** after they are assigned, and who practice course concepts as much as possible.
  - It is better to ask a question sooner than later -- for example, it is better to send an e-mail with a specific question as soon as you think of it than it is to wait a day or two until the next class meeting or office hour. If you wait to ask such questions, you may not have time to complete the assignment.
  - It is perfectly reasonable if you send me a question and then end up finding out the answer yourself before you receive my answer; likewise, it is not a problem if you end up sending me several questions in separate e-mails (as you work on different parts of a homework while awaiting earlier answers).
- That said, I am expecting that you will ask **specific** questions – overly vague or broad questions are problematic. (For example, an example of a specific question is, "When I try to compile function X, I receive the following error message: … and my code is attached to the end of this e-mail. Can you point me in the right direction about what is wrong?" An example of an overly vague or broad question is: "Here's my code. Is it right?"
- I try to check my e-mail (st10@humboldt.edu or sharon.tuttle@humboldt.edu or smtuttle@humboldt.edu) about once a day on weekdays, and about once over each weekend. This is another reason to start assignments early, so that you have time to receive a reply to any questions that might arise. Include **CS 131** and a general description of your topic in the Subject: line, both because including this makes it less likely that I'll overlook your question, and because it will make your message stand out if it the spam filter gets confused and puts it in the university spam quarantine.
  - If I have not replied to your e-mail within 24 hours, please re-send it, just in case it did get overlooked somehow.

- Also, try to avoid the word "password" in your e-mails to me -- pwd is a handy abbreviation to use instead -- because, due to phishing scams, HSU's spam filtering definitely does not like e-mails with that word in it! (Odd, but this was definitely the case in Spring 2010...)

Additional Homework-Related Policies:

- You should not expect to be able to finish homework assignments during the lab sessions--- like any college-level course, you should expect to put in a (potentially) large amount of time outside of scheduled class meetings (lectures and labs) doing the assigned reading, working on homework assignments, and practicing concepts discussed.
- Each homework must be submitted as is specified on the homework handout in order to be accepted for credit. This may vary for different homeworks. Often, parts of homeworks will be submitted using a special tool on nrs-labs. Code that does not run in DrRacket or on nrs-labs will not receive credit; remember that it is your responsibility to verify that your code runs in DrRacket or on nrs-labs for each homework before submitting its code, regardless of where you developed that code.
- Each homework will be clearly marked with one or more due dates (a single homework could have multiple parts with multiple due dates).
  - **No homework will be accepted late. If you wish to receive any credit for a homework, then you must turn in whatever you have done, even if it is incomplete, by the deadline. Partial credit is usually preferable to no credit.** Note that "the computer/network/etc. going down" is no excuse --- if you leave a homework for the last minute and there are technical problems, you still must turn in whatever you have by the deadline. As with any work done on computer, make frequent back-ups of your files!
  - You may submit multiple versions of homework files before the deadline; I will grade the latest pre-deadline submission unless you inform me otherwise. This is to encourage you to turn homework parts in early (since you will know that you can always turn in an improved version if further inspiration strikes). You also don't have to worry about forgetting to submit something that has already been submitted.
- The tool that you will be using to submit homework parts results in a file that serves as your "receipt" for having submitted items. You are expected to retain these "receipt" files at least until a grade has been posted to the course Moodle site for that homework. If there is a system glitch or other hardware/software/network problem, you may be asked to make me a copy of one or more receipt files; if you do not have them, then you will not receive credit for the files involved. These receipt files are for your protection!
- It is nearly impossible to write unambiguous computer program specifications. If you have questions about "what she means", get them resolved very early in the development cycle by **asking**.
- There is more to computer programs than simply whether they run or not...
  - Part of your homework grade will be determined by how well your programs meets the written requirements. Programs are expected to meet homework handout specifications precisely; when one eventually works within a team on large software projects, following the specifications precisely is vital, and can mean the difference between a working product and one that just sits there.
  - Note that programs may be graded on **style** as well as on whether they run properly and whether they precisely meet the homework specifications and requirements. Discussions on programming style will be ongoing throughout the semester.
  - Likewise, because you will be learning good problem-solving practices in this course as well as

programming syntax, part of your homework grade will be based on whether you are following these practices (following the design recipe, including required documentation and tests, etc.)

- Some homework problems (and lab exercise problems) may be graded simply based on whether they have been attempted (the instructor's decision is final as to whether this is the case) --- other problems may be graded for correctness, style, and whether they meet specifications. You will not know in advance which will be the case.

Additional Grading-Related Policies:

- If you would like me to e-mail certain course grades to you during the semester, then you must give me permission in writing on the course information form.
- Clicker questions will be given during most lectures and labs (although probably more clicker questions will be asked during lecture sessions); graded lab exercises will be given during most lab sessions. The two lowest lab exercise grades will be dropped from your grade. Between the ample quantity of clicker questions and the dropped lab exercise grades, then, you can be absent several times from non-exam lecture or lab sessions without direct penalty, for whatever reason (although you are, of course, still responsible for the material covered on those days, and it is your responsibility for determining what that material is).
- Note: **NO** homework grades are dropped; **ALL** homework grades count toward your homework average. Every homework includes important practice of programming fundamentals.

Additional Course Policies:

- You are expected to read this syllabus and be prepared to sign a statement that says you have received and understand these policies.
- Exam dates are given in the course schedule below. Make-up exams are only possible by special prior arrangement, or because of a valid medical excuse.
- You should monitor your e-mail for course-related messages. The University provides a means for you to specify your preferred e-mail address, so if you wish to receive e-mail into an account other than the one HSU provides, change your preferred e-mail address in both Account Center and Moodle accordingly. Course-related messages from me will include **CS 131** in the Subject: line.
- You are expected to check the public course web page and the course Moodle site regularly --- course handouts, homework assignments, example code from lectures, and possibly more will be posted to the public course web page, and grades will be posted to the course Moodle site. You are expected to monitor your posted grades and let me know about any discrepancies.
- When reading assignments are given, you are expected to prepare (read and study) assigned readings before class and to participate in class discussions. Projected examples will be utilized frequently during discussion. You should understand that there may be material in the reading that will not be discussed in lecture/lab, and material in the lectures/labs that may not be found in the reading. You are responsible for both.
- **Attendance and disruptive behavior:** Students are responsible for knowing policy regarding attendance and disruptive behavior: http://www.humboldt.edu/studentrights/attendance_behavior.php
- Regular attendance at lecture and lab sessions is expected. If you should happen to miss a lecture or a lab, then you are responsible for finding out what you missed. "I wasn't there that time" is never an acceptable excuse. Lecture notes are not posted, although many of the projected examples will be

made available on the public course web site. Clicker questions and graded lab exercises missed **cannot** be made up later.

- As previously mentioned, during lab sessions, there may be lab exercises due during that lab session. Once a lab's lab exercise is complete, the remaining lab time should be used to continue work on the current course homework assignment, to practice course concepts, and/or to ask questions about course-related topics.
- **Late arrival to class:** Please attempt to come to class on time, with your headphones put away and your cell phones turned off. If you must arrive late or leave early, please do so with the least possible distraction to other students. If your late/early habits become disruptive, you may be asked to leave the class permanently.
- **Class disruption:** University policy requires that instructors eliminate disruptions to the educational process. Distractions such as excess talking, ringing cell phones, working on assignments for other classes, inappropriate or distracting laptop/tablet/smartphone/gadget use, demonstrations of affection, packing of books early, loud music leaking from headphones, chronic late arrivals or early departures, excessive comings and goings or other behaviors that disrupt the class are not acceptable. Students indulging in such behaviors will first be warned before being required to leave the class permanently.
- **Emergency Evacuation:** Please review the evacuation plan for the classroom (posted on the orange signs), and review the campus Emergency Preparedness web site at: http://www.humboldt.edu/emergencymgmtprogram/campus_emergency_preparedness.php for information on campus Emergency Procedures. During an emergency, information regarding campus conditions can be found at **826-INFO** or: http://www.humboldt.edu/emergency

Tentative Course Schedule: (subject to change!)

Week 1: August 24, 26, 27

- Topics: Course introduction; syntax and semantics; simple and compound expressions; types (number, boolean, string, image); named constants
- Reading: from How to Design Programs, 2nd Edition (HtDP/2e): Section 1 Prologue: Introduction, and Sections 1.1, 2.1
- HW #1 out

Week 2: August 31, September 2, 3

- Topics: Introduction to designing your own functions and to the program design recipe; parameter variables
- Reading: HtDP/2e: Sections 1.2, 2.2, 2.3
- HW #1 due, HW #2 out

Week 3: September 7, 9, 10

- (Monday, September 6 - Labor Day Holiday - does not affect CS 131)
- Topics: Boolean functions, conditional expressions, conditional functions, enumerations, intervals,

and itemizations, adding the template step to the program design recipe

- Reading: HtDP/2e: Sections 1.3 - 1.7, 2.4
- HW #2 due, HW #3 out

Week 4: September 14, 16, 17

- Topics: Introduction to structures
- Reading: HtDP/2e: Section 2.5
- HW #3 due, HW #4 out

Week 5: September 21, 23, 24

- Topics: Introduction to lists
- Reading: HtDP/2e: Section 4.1, 4.2, 4.3.1
- HW #4 due, HW #5 out

Week 6: September 28, 30, October 1

- Topics: More on lists
- Reading: HtDP/2e: Sections 4.3.2, 4.3.3, 4.3.4
- Friday, October 1st: Review for Exam #1
- HW #5 due

Week 7: October 5

- Tuesday, October 5: EXAM #1
- NO CLASS Thursday, October 7, and Friday, October 8 - instructor at a conference

Week 8: October 12, 14, 15

- Topics: Introduction to C++; functions in C++; conditional statements in C++
- Reading: HtDP C++ Transition readings on Moodle: Sections 1 - 4
- Optional Reference: Savitch, Section 2.4, pp. 74 - 84, Section 3.2
- HW #6 out

Week 9: October 19, 21, 22

- Topics: From Racket structures to C++ classes
- Optional Reference: Savitch, Section 10.2
- HW #6 due, HW #7 out

Week 10: October 26, 28, 29

- Topics: Mutation; assignment statements; the ++ and += operators; local variables; scope; zero-argument constructors, modifier and "other" methods, overloaded methods; pass-by-value
- Reading: HtDP C++ Transition readings on Moodle: Sections 5.1, 5.2
- Optional Reference: Savitch, Sections 2.1, 10.2, 4.3
- HW #7 due, HW #8 out

Week 11: November 2, 4, 5

- Topics: Introduction to `cout` (screen output) and `boolalpha`; introduction to (mutation-based) repetition - count-controlled loops, while-loops, for-loops; introduction to arrays
- Reading: HtDP C++ Transition readings on Moodle: Section 5.5
- Optional Reference: Savitch, Section 2.2, pp. 50-55, Section 2-4, pp. 84-93, Sections 3.3, 7.1, Section 7.2, pp. 385-391
- HW #8 due, HW #9 out

Week 12: November 9, 12

- Topics: Definition of a C++ program; writing a `main` function; compiling and linking C++ functions to create a C++ program; `void` functions
- NO CLASS Thursday, November 11 - Veterans Day Holiday
- Reading: HtDP C++ Transition readings on Moodle: Sections 5.3, 5.4, 5.6, 5.7
- Optional Reference: Savitch, Section 5.1, Section 1.3, pp. 24-29
- Friday, November 12th: Review for Exam #2
- HW #9 due

Week 13: November 16, 18, 19

- Tuesday, November 16: EXAM #2
- Topics: Introduction to `cin` (interactive input); sentinel-controlled loops; functions that modify argument arrays
- Reading: HtDP C++ Transition readings on Moodle: Section 5.5
- Optional Reference: Savitch, Section 2.2, pp. 56-59, Section 7.2, pp. 391-394
- HW #10 out

Thanksgiving Holiday - Monday-Friday, November 22-26

Week 14: November 30, December 2, 3

- Topics: Introduction to pointers; pass-by-reference; dynamic memory allocation; intro to C++ file

input/output

- Optional Reference: Savitch, Sections 5.2, 9.1 9.2, 6.1
- HW #10 due, HW #11 out

Week 15: December 7, 9, 10

- Topics: Introduction to linked lists; the "big three" needed when a C++ class includes dynamic memory allocation
- Optional Reference: Savitch, Sections 11.4, 13.1
- Friday, December 10th: Review for Final Exam
- HW #11 due

Final Exam:

THURSDAY, December 16, 3:00 - 4:50 pm, in GH 218 (unless I announce otherwise)