Welcome to the class notes for CS253 (fall semester, 2015). Although it may surprise the more cynical among you, professors are not just “winging it” when we give lectures. Before every class, we write out notes on what we intend to say. The notes you will see this semester originate with Bruce Draper, who has taught this course multiple times and has considerable experience with this material. However, what you will also see as the course moves forward is the blending of material from both Bruce Draper and myself. There is nothing new, nor unusual, about multiple professors working together over many semesters to develop and mature the material for a course. Just for example, CS 410 is typically shared between Bruce Draper and myself and has been for nearly two decades. This will be my first time through CS 253 and I can already predict you will at times benefit from my fresh approach and a newcomers awareness of what is hard versus easy. There will also, no doubt, come moments where both you and I may think fond thoughts of having direct access to Bruce Drapers years of experience with this material.

In addition to typically handing out notes after a lecture, I will be adding my own PowerPoint summaries as well as new code example to be run in lecture. From the standpoint of presentation, this is one way in which I will be bringing something new to the way CS 253 is taught. Now a comment about these notes and what they are and aren’t good for with the expectation that will help you make best us of them.

Firsts, this should be obvious, the notes do replace the textbook! Your primary text for this class is C++ for Java Programmers by Mark Allen Weiss and it is important that you read this book. It provides the overall structure for the class, and describes many of the differences between C++ and Java in detail. Moreover, I do not generally repeat the material from the book in lectures. That would be redundant. The lectures emphasize what is important in the text, fill in holes, update the text where is it outdated, and occasionally disagree with the text, but they do not repeat it. If you try to use these notes as a replacement for the text, it will hurt you. Besides, the text really isn’t very long, so the reading assignments shouldn’t be too taxing.

The lectures go beyond the text in many ways. The text, to be blunt, is useful but shallow. The important topics in this class, e.g. memory management, styles of object orientated programming, and how C++ is implemented “under the hood”, are only discussed superficially by Weiss. The lectures expound on these topics much more thoroughly. Indeed, we go will through seven weeks where there are no reading assignments in the book; all the material in this part of the course comes from the lectures.

The lectures for this class are video taped available through Canvas. That noted, I strongly recommend coming to the lectures in person. Not only does it allow you to ask questions and to hear the questions
of other students\textsuperscript{1}, but psychologically it is more compelling. You will remember lectures better if you were there. Nonetheless, if you can’t make a particular lecture in person, the lectures are available online. Remember, you are responsible for all material presented in class. A bit of experience: this class has been taped in past semester. Overall, students who came to the lectures did better than those who didn’t. That outcome should not come as a surprise to anyone.

When attending or watching a lecture, I recommend taking notes. Even if you never study your notes, the act of taking notes during class makes you think about and integrate the material in a way that leads to learning. Besides, these notes reflect what I intended to say, not what I did say. Often lectures are modified on the fly in response to a student question (or simply because I see that puzzled “I am lost” expression in the audience). This is particularly true in terms of how the lectures relate to the homework. These notes are not a good replacement for taking your own.

What these notes are good for is a study aid before midterms and finals. They should remind you of what was discussed in class, and if you take your own notes, you will find that sometimes your notes aren’t clear, and sometimes these notes aren’t clear, but together the information you need is there. If not, they indicate topics you might want to ask about in class or during office hours.

\footnote{Students, unlike the professor, do not have microphones. I try to repeat the question, but I often forget.}
Welcome to CS253. I am Prof. Ross Beveridge, and I will be teaching this class. You should also get to know your two Graduate Teaching Assistants (GTAs): Fereydoon Vefaie and Hao Zhang.

Let me start with some quick, procedural notes. This is a Tuesday Thursday class, so the lectures are one hour and fifteen minutes long. While that might seem to suggest we have time for a relaxed start, no. To get through the material we need to cover means students being on-time and us getting started at 9:30 prompt. It should also be said that an hour and fifteen minutes is a long time for you to maintain interest and for me to keep the flow of information active, worthwhile, and two-way. I will do my best and I am counting on your help.

As you may have noticed, this is a large class, with over 100 students registered for it. To make things worse, I am truly bad at names. I can barely remember my own kid’s names sometimes. So how am I going to learn all your names? I won’t, but I will learn as many as I can. To help me, I will ask you your name whenever you speak in class, whether you are asking a question or I called on you.

The interesting part of today’s lecture is why? Why are you here? But you have a quiz due before class on Friday, so I have to get to the administrative aspects of the class first. So let me start with what, when and how much before we get to why...

Most of the important information about class activities area available through the public facing course website: [http://www.cs.colostate.edu/~cs253/](http://www.cs.colostate.edu/~cs253/) Please turn first to this site and secondarily to the Canvas site associated with the course. Note the key difference between the two is that anything remotely public goes on the public facing site. Matter clearly private, grading and quizzes come quickly to mind, live behind the Canvas firewall and tied directly to your identity as a student in this course.

This course has reading assignments (with quizzes), recitations, programming assignments and tests. Let’s go over these:

- **Reading assignments & Textbooks**
  - The Weiss text is mandatory. This is what many of the lectures are structured around.
  - The Stroustrup text is optional – but I use it as a reference. The language specification is available on the web, but Stroustrup provides more explanations. If you use the web instead, please use [www.cplusplus.com](http://www.cplusplus.com) (you can find the link on the resources page of the class web site).
  - Reading assignments are posted on the web (progress page). They are listed by the day they are due. For example, by Thursday you need to have read Chapter 0 of Weiss.

- **Quizzes**
- There will be quizzes on the reading. Why? If you do the readings on time, you will understand the lectures better and do better on the tests and home-works. The lectures do not recap the readings, they expand on them. The quizzes are designed to be relatively easy, if you do the reading.
- The quizzes are on-line via Canvas. You may take a quiz only once, and you have 15 minutes.
- The quizzes become available after the lecture in which they are assigned, and may be taken until 30 minutes before the start of the class the day they are due. For example, there is a reading assignment for Thursday (Chapter 0), so there is an on-line quiz that will appear after today’s class, and which you must start before 9:00AM on Thursday.

- **Lectures** – you are responsible for all the material presented in lectures, some of which is not in the book. (Mandatory)
  - Lectures may or may not use PowerPoint. Even if there are PowerPoint notes, they are sparse at best. They are no substitute for your own notes! Pay attention! ask questions!
  - If you will miss a lecture, make sure someone takes notes for you (you are still responsible for the material)
  - Lectures begin at 9:30, not 9:30 or later. That means you are at your desks ready to go at 9:30, not walking through the doors of Clark at 9:30. “To be on time is to be late; to be early is to be on time.” Have respect for the class, your peers, and me.
  - I will end class close to 10:45 in return.
  - Put your cell phones in silent mode; if you get an emergency call, leave the room before answering.
  - The lectures are being recorded. They will be posted to Canvas as a study aid, but I strongly recommend attending in person. That way you can ask questions, hear the questions of others, and in general be more engaged.
  - Participation is required – students will be called upon to actively engage with questions during lecture. This is good for many reasons, including: it helps you learn to speak in groups and think on your feet, it helps me keep better touch with the students. The use of the “Pelican” will be explained in lecture as a means of organizing the flow of questions and answers. Finally, one of the most important answers for the instructor to hear may be: “I am truly confused and don’t really know how even to approach what you are asking me.” If that is how one student is responding, odds are very high that they are not alone.

- **Recitations**
  - No recitations this week, recitations start next Monday.
    - Otherwise, recitations are mandatory
    - Except for the weeks of midterms and the final
    - Those weeks, recitations are optional.
  - Generally will be taught by one of the GTAs.
  - Recitations do not recapitulate the lectures. They present tools necessary for software development under unix.
    - Unix, debuggers, profilers, unit test frameworks, etc.
• An exercise is due at the end of most recitations

• Programming assignments – approximately 9 of them, spread over 11 weeks (your assignments may vary).
  o They build on each other – the 2nd is to extend the 1st, etc. You can’t skip one.
    ▪ No, I won’t tell you what it coming ahead of time
  o I will tell you what you program should do, but not how to do it. That is your task.
    (Although the lectures will give you hints.)
  o They are due when they are due. No credit for late assignments
    ▪ And they build on each other, so if you miss one, you’ll have to do it anyway.
    You just won’t get credit for it.
  o Some assignments are two-week assignments, others are one-week assignments. Two-week assignments are worth twice as much as one-week assignments.
  o Automatically graded: we build test data sets, we run your program, we then compare results to true results.
    ▪ If it doesn’t compile, 0 points
    ▪ Assignments may have special instructions
      • On some, compiler warnings deduct points (up to 10%)
      • On some, memory issues detected by valgrind will deduct points
      • One assignment will be graded on speed
    ▪ Testing your code is your responsibility. Think of difficult test cases. Try them.
      Never trust the user or a file – make up bad inputs, make sure your program doesn’t crash.
    ▪ If your program crashes on an input file, you get no points for that case (even if it produced the right answer first).
    ▪ We will release the test set so you can double check our tests
      • If you find a discrepancy, talk to the GTAs
      • If you disagree with policy, talk to me
    ▪ What about documentation? Style? We don’t have to grade for these. The assignments build on each other. After 3 months, these are self-rewarding. Trust me.
  o Compiled and graded ON DEPARTMENT MACHINES. It doesn’t matter if it worked on your home laptop...
  o My laptop crashed is not an excuse. You have access to department machines, and they are backed up nightly. If you choose not to use them, you take responsibility...
  o Generally, CS253 is a ‘no excuses zone’. Employers are not supportive of prospective or current employees who provide reasons they failed to meet expectations. Therefore, consider CS253 an excellent opportunity to practice those basic professional skills that will help to get you and keep you employed in work you find interesting and challenging.

• Exams
  o 1st midterm (Thursday October 1st) : 10% of your grade
  o 2nd midterm (Thursday November 5th) : 10% of your grade
Final exam: 20% of your grade (Thursday December 17th at 6:20PM)
I reserve the right to change the midterm dates with one week’s notice (in case we fall behind or get ahead as a class).
Warning: I give hard tests. The mean or the raw score is often close to 50%, with a large st. dev. This is how I get a true reading on what you know, and make sure that one “stupid mistake” doesn’t torpedo your grade. But it can be demoralizing. Don’t give up or get flustered: the scores will be curved.

• Grading
  Your overall score calculated as:
  • 40% programming assignments
  • 40% tests
    • 10% each midterm
    • 20% final
  • 10% quizzes
  • 10% recitations
Tests are curved (they are very hard). Other things may be curved, at my discretion.
To get a C or better, you need:
  • Overall score of 70% or better
  • At least 65% on programming
  • At least 65% on tests
I reserve the right to apply a curve at the end, but may not.

• Resources
  Your text (Weiss)
  References (Stroustrup or www.cplusplus.com)
  The CS253 class web site (www.cs.colostate.edu/~cs253)
    • Important news on home page
    • Progress page provides links to lecture notes
    • Assignments on assignment page
    • Recitations on recitation page
    • Resources page has links to additional resources
  Canvas
    • Keep track of your grades
    • Take on-line quizzes
    • Links to videos of lectures.

• Academic Integrity
  I hate this topic. Most of you are honest and won’t have any problems. But...
  All programming assignments, recitation assignments, quizzes and tests in this course are solo projects. They may not be done in teams. You may not copy code from the internet. You may not submit any work that is not your own. Ever.
  We will actively look for cheating. No, I will not tell you how.
The department and university policies on academic integrity will be strictly enforced, and all cases reported to the university. If you are unclear about these policies, there are links on the class website (syllabus page).

That includes the rule that you are responsible for making sure that no one cheats off you. You are responsible for keeping your own work secure.

Does this mean you can’t talk with your classmates? No. It means you can’t write code for them, or they for you. Abstract discussions are fine.

When in doubt, ask me.

• **Unforeseeable Emergencies**
  - The rule about no late assignments has one exception: unforeseeable emergencies.
  - Examples include: death in family, illness (usually requiring hospitalization), house fires, etc.
  - This probably won’t happen to you … but there are over 100 of you. Unfortunately, there is a significant likelihood that it could happen to someone in this room. It happened to me as a freshman. If it does happen to you, talk to me. Tests can be rescheduled, assignments can be delayed, etc., if the emergency justifies it.

Assignment: Read Chapter 0 (yes, 0) of Weiss for Thursday. There is an on-line quiz that closes at 9:00AM Thursday morning.

Now let me open a major question anyway: Why are you here? Why are you taking this class? No wait… many of you are thinking “because its required”. That’s true, but it only pushes back the question: why have the faculty made this a required course? What should you learn? What do we want you to learn?

Here are some reasons:

• **To learn C++**.
• **To learn a 2nd language (and how to learn a second language)**
• **To bridge the gap between CS200 & CS270**.
• **To learn what is going on “under the hood”**.
• **To learn more about Java!**
• **To learn how to select a language**
• **To learn about memory management**.
• **To learn to write efficient and/or real-time programs**.
• **To learn (through experience) how to write re-usable code**
• **To learn tools for developing code in unix**
• **To learn to write, maintain and test large programs**