

Introduction To CPSC 217

James Tam

Administrative (James Tam)

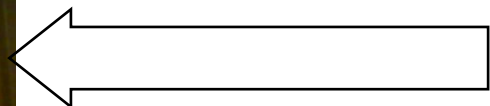


- Contact Information

- Office: ICT 707 
- Email: tamj@cpsc.ucalgary.ca

- Office hours

- Office hours: Mon 13:00 – 13:50, 15:00 – 15:50, Wed 11:00 – 11:50
- If I'm not in my office give me a few minutes or check the lecture room.
- Email: (any time)
- Appointment: email, phone or call
- Drop by for urgent requests (but no guarantee that I will be in if it's outside of my office hours!)



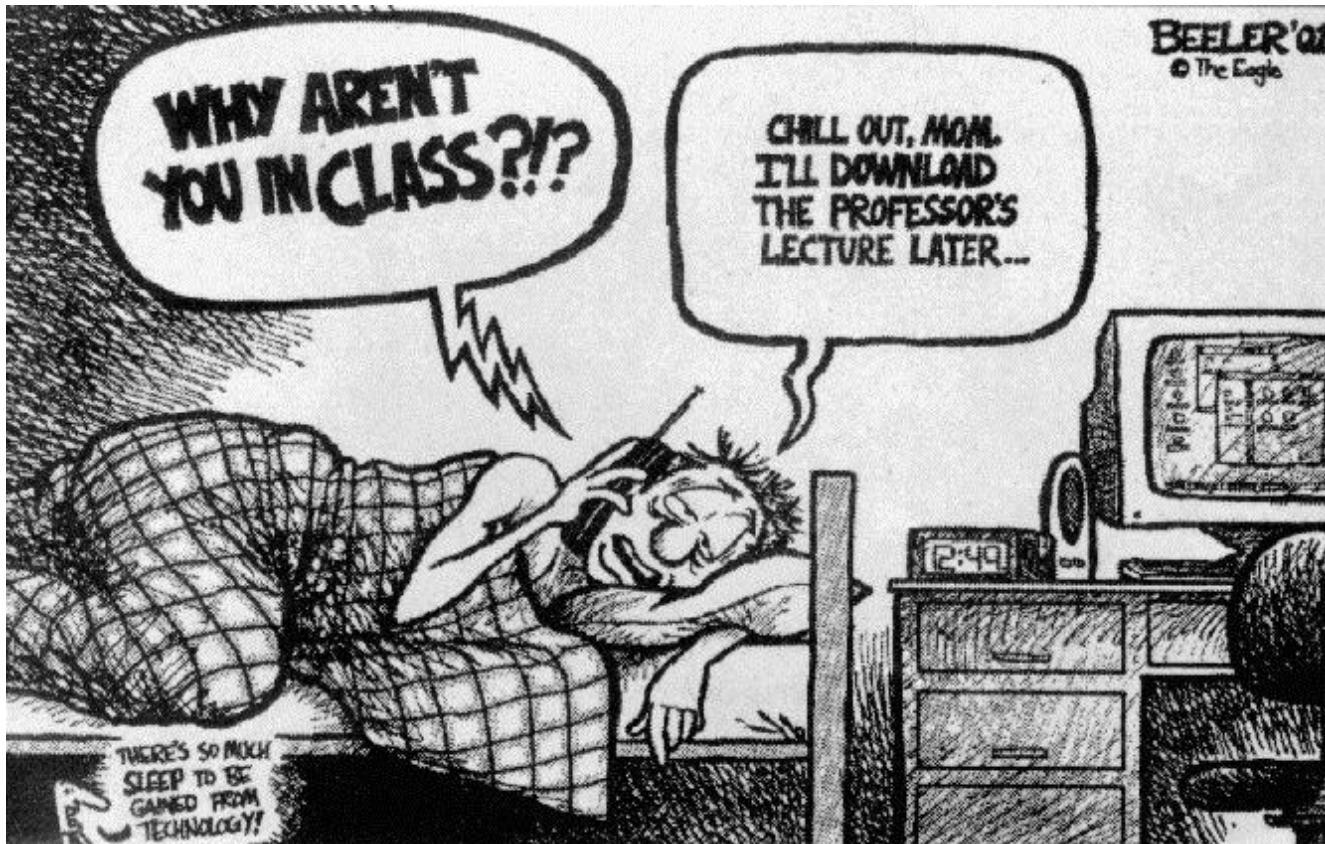
←    My Office

Course Resources

- Required resources:
 - Course website: <http://pages.cpsc.ucalgary.ca/~tamj/217> (Get the notes off the course webpage before lecture)
- Recommended but not required:
 - "Starting Out with Python"(Gaddis T.) Addison-Wesley.
 - Alternatively you can access any book licensed by the university (“for free”) on the library web site:
 - (One of many books available)” Visual QuickStart guide (2nd Ed)
<http://proquest.safaribooksonline.com.ezproxy.lib.ucalgary.ca/>

How To Use The Course Resources

- They are provided to support and supplement this class.
- Neither the course notes nor the text book are meant as a substitute for regular class attendance.



How To Use The Course Resources (2)

```
def display (world):

    sys.stdout.write(' ')
    for i in range (0, columns, 1):
        if (i < 10):
            print i,
        else:
            num = i + 55
            ch = chr(num)
            print ch,

    print

    for i in range (0, columns, 1):
        sys.stdout.write(' -')
    print

    for r in range (0, rows, 1):
        for c in range (0, columns, 1):
            sys.stdout.write('|')
            sys.stdout.write(world[r][c].appearance)
        print ('|'), r
        for i in range (0, columns, 1):
            sys.stdout.write(' -')
    print
```

How To Use The Course Resources (2)

```
def display(world):  
    sys.stdout.write(' ')  
    for i in range(0, columns, 1):  
        if i < 10:  
            print i,  
        else:  
            num = i + 55  
            ch = chr(num)  
            print ch,  
    print  
  
    for i in range(0, columns, 1):  
        sys.stdout.write(' -')  
    print  
  
    for r in range(0, rows, 1):  
        for c in range(0, columns, 1):  
            sys.stdout.write('|')  
            sys.stdout.write(world[r][c].appearance)  
        print ('|'), r  
        for i in range(0, columns, 1):  
            sys.stdout.write(' -')  
    print
```

If you miss a class make sure that you catch up on what you missed (get someone's class notes)

...When you do make it to class make sure that you supplement the slides with your own notes (because you aren't going to remember it in the exams if you don't)

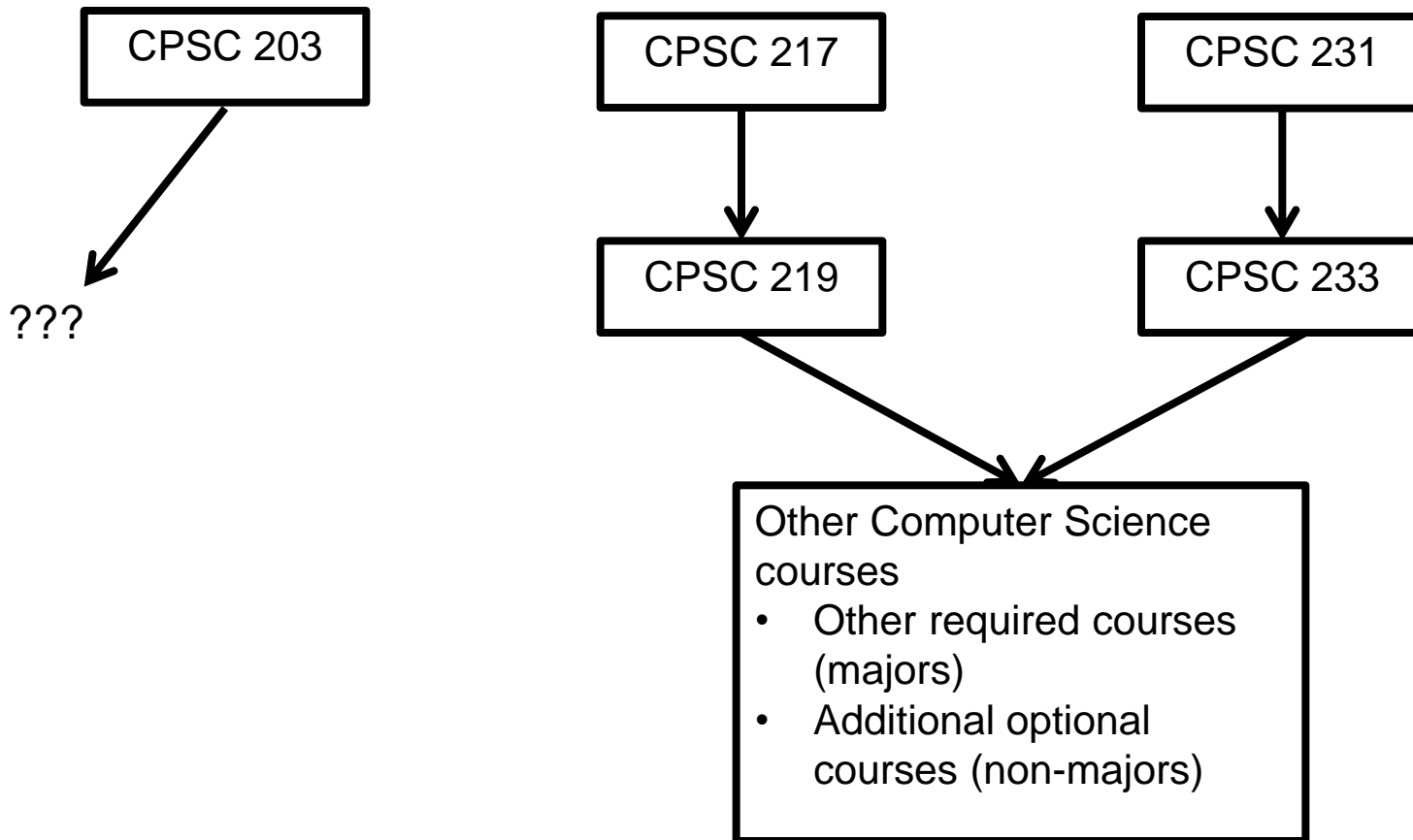
How To Use The Course Resources (3)

- What you are responsible for:
 - Keeping up with the content in class which includes the topics covered but also announcements or assignments whether you were present in the class or not.
 - If you are absent, then you are responsible for getting the information from the other students in class.
 - (I won't be able to repeat the lecture content if you are absent...there's just too many of you to make it practical).
- However, after you've caught up by talking with a classmate:
 - Ask for help if you need it
 - There are no dumb questions



Are You In Right Place?

- There are three main introductory Computer Science classes.



Common Computer Skills Assumed

- You know what a computer is!
- You've used a computer in some form (e.g., turn on, turn off, open a file, gone online etc.)
- You have experience with the simple features found in commonly used applications (specifically email, web browsers, text editing using a word processor).

What This Course Is About

- Writing/creating computer programs.
- But it is not assumed that you have prior knowledge of Computer Science.
- It can be a lot of work.



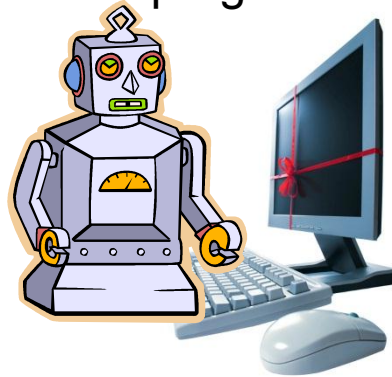
- The course can be completed by students with a normal course load (many already have gotten through it!)
- But be cautious if you already have many other commitments

But Learning How To Write A Program Is Only A Part Of Creating A Program.

- When creating a program there isn't an exact prescribed formula or series of steps that you can learn and apply.



'Self' programming computers not possible



- But you aren't left alone to fend for yourself!



Common Interview Questions

- Besides looking at degrees granted and grades received, many tech companies (e.g., Google) may ask you questions that appear non-technical:
 - You're asked to solve puzzles during the interview.
- There is a relationship between skill at solving puzzles (“problem solving”) and success in a (technically oriented) industry.

Example list of questions

<http://www.businessinsider.com/15-google-interview-questions-that-will-make-you-feel-stupid-2009-11>

This Why Computer Science Is About Problem Solving

- A simplified description of what ‘problem solving’ means:
Write a computer program that performs a task (fulfilling a need and thus solving a problem).
- This requires that you know how to write a program in a given language but goes beyond knowing the rules and structure of a language (this is the problem solving aspect...how do you *apply your knowledge and skills*).
 - Analogy: you may know the rules and structure required to produce a poem but it takes more than that to write good poetry.
 - “Ode to my keyboard”
- For example you may know how to get a program to run across the Internet but you may not know how to write a good game app. on Facebook™.
 - “*This *%\$#! App really sucks!*”

Computer Science Is About Problem Solving (2)

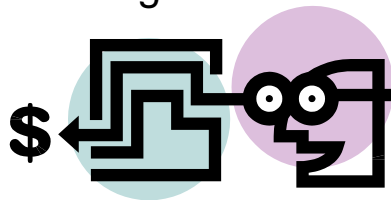
- Also while you may be able to develop a ‘working solution’, that solution may be a very poor one according to different (but important criteria):
 - Bloated solution: a better program could have been written in a fraction of the current size.
 - Inefficient solution: the series of steps results an slow running program (it’s possible to write solutions so the program runs faster).
 - Cryptic/hard to maintain solution: the program is written in a way that makes it hard to understand. This results in many problems not the least of which is program maintenance (“I don’t want to touch this guy’s code because I’m afraid of breaking something”).
 - Etc.
- You get better at problem solving through practice - “How to succeed in this course” (coming up).
 - This is why lectures won’t directly address the solution to an assignment.

Computer Science Is About Problem Solving (3)

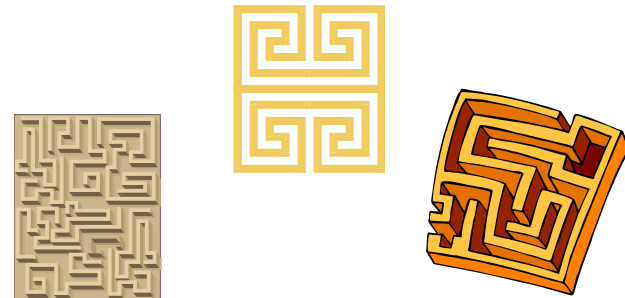
- You will be taught:

- Computer programming: The mechanics of how different programming concepts work e.g., How do you get a program to read from a file.
- Problem solving strategies: approaches to creating a solution to a challenging problem:
 - Practice! Practice! Practice!
 - Example strategy: Problem decomposition.
 - Example strategy: Visualization techniques (What does the problem entail?)
 - Good programming style.

Providing solutions to assignments may be popular among students but useless for learning



What's needed is for me to teach you the skills to solve any reasonable problem



Course Goals

- Understand basic programming constructs (‘concepts’) such as branching and looping.
- Develop basic problem solving and analysis skills.
- Being able to implement a solution for a moderately sized problem using good design principles.

How To Succeed

- Successful people



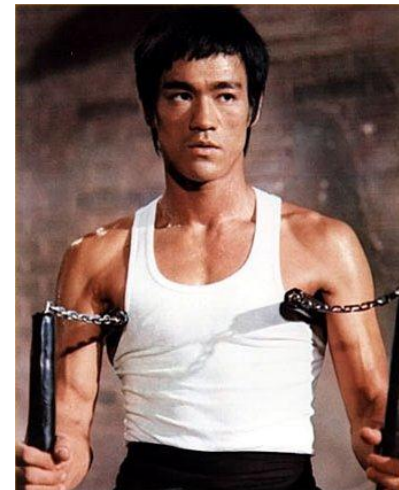
Leonardo Da Vinci



J.R.R. Tolkien



Amadeus Mozart



Bruce Lee

How To Succeed In This Course

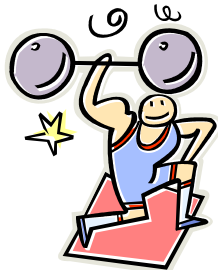
1. Practice things yourself.

- “I wish he [JT] would help us more by giving us code [parts of a computer program] that can be directly used in the assignment.”

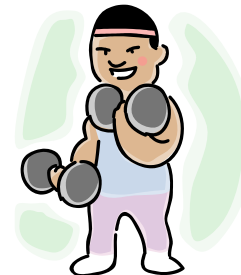


- How Computer Science works: You get better by doing things for yourself (this is a ‘hands-on’ field of study and work).

Similar to getting fit: you can't just watch



You have to do it yourself



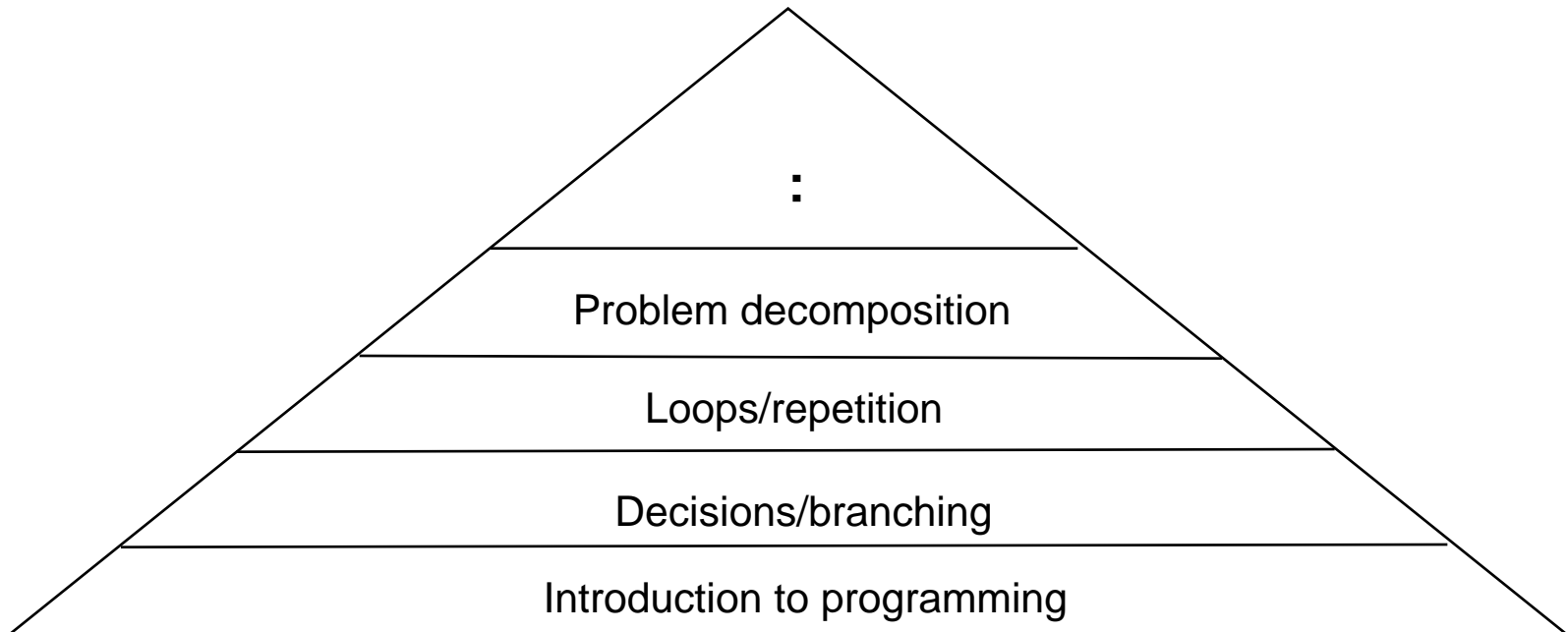
How To Succeed In This Course (2)

- Write lots programs.
 - At the *very least* attempt every assignment.
 - Try to do some additional practice work (some examples will be given in class, some practice assignments will be available on the course web page).
 - Write lots of little ‘test’ programs to help you understand and apply the concepts being taught.
- Trace lots of code (computer programs)
 - Reading through programs that other people have written, and executing it ‘by hand’ in order to understand how and why it works the way that it does.
 - This is an essential skill.

How To Succeed In This Course (3)

2. Make sure that you keep up with the material

- Many of the concepts taught later depend upon your knowledge of earlier concepts.
- Don't let yourself fall behind!
- *At least* attempt all assignments!



How To Succeed In This Course (4)

- If you find concepts unclear trying to understand them on your own can be beneficial (because this is a ‘hands on’ field).
 - Read alternate explanations of the concepts covered in class in the text book (or other textbooks: remember that electronics books accessible through the library (Safari) are ‘free’).
 - Looking at online resources:
 - Remember academic resources just like other online information may not be a good source.
 - Start with more reputable sources e.g., www.python.org
- Addendum to the previous point #2 and a point raised earlier “ask questions”.
 - If you are still unclear on concepts then make sure that you ask for help.
 - Don’t wait too long to do this because latter concepts may strongly depend on the understanding of earlier concepts.
 - (If your first time that you come for help is the last week of the term or worse after the end of term then it’s probably far too late).

How To Succeed In This Course (5)

3. Look at the material before coming to lecture so you have a rough idea of what I will be talking about that day:
 - a) Read the slides
 - b) Look through the textbook(s)

How To Succeed In This Course (6)

4. Start working on things as early as possible:
 - Don't cram the material just before the exam, instead you should be studying the concepts as you learn them throughout the term.
 - It's important to work through and understand concepts **before** you start assignments. If you try to learn a new concept and work out a solution for the assignment at the same time then you may become overwhelmed.
 - Don't start assignments the night (or day!) that they are due, they may take more time than you first thought so start as soon as possible.

How To Succeed In This Course: A Summary

1. Practice things yourself
2. Make sure that you keep up with the material
3. Look at the material before coming to lecture
4. Start working on things early

Computer Science: Labs And Tutorials (Reminder)

- Labs (“Continuous Tutorial/CT”):
 - Attendance is not required (no official registration)
 - Q & A session: it will be used as an additional place where you can get help.
 - The CT schedule will be posted early in the semester.
- Tutorials:
 - They will be conducted by the Teaching Assistants (TA).
 - A mandatory component of the course (registration in a specific section is required).
 - Quizzes will be written during some tutorials (see the schedule):
 - http://pages.cpsc.ucalgary.ca/~tamj/217/#Assignments_and_exams
 - Review of concepts covered in lecture (especially some of the more challenging ones).
 - Discussion of assignment requirements.

Computer Science: Labs And Tutorials (Reminder: 2)

- (Tutorial information continued):
 - Practice exercises.
 - ‘Open tutorials’ will sometimes be held (extra CT time where your TA will be available to help students).
- More information about tutorials and labs is available on the course web site:
 - http://pages.cpsc.ucalgary.ca/~tamj/217/#Tutorial_and_lab_Information

Evaluation Components

- Six quizzes (5%)
- Five assignments (30%)
- Two examinations (65%)
 - Midterm = 30%
 - Final exam = 35%

Quizzes

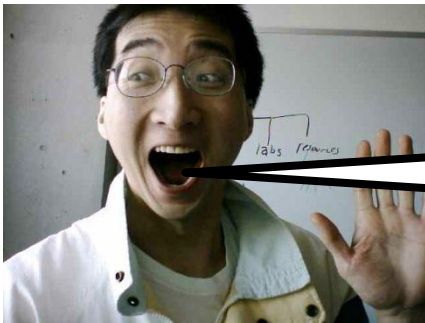
- They will be conducted in tutorial (one reason why that component is mandatory and you had to register for a specific one).
- Completed by hand on paper (extra and early practice for the examinations).
 - **DON'T FORGET TO FILL IN YOUR IDENTIFYING INFORMATION AND HAND IN YOUR QUIZ BEFORE LEAVING THE ROOM.**
 - **(This applies to the exams as well...)**
 - (Just like the exam: once you are 'out the door' it's too late).
- Although quizzes may involve describing concepts or tracing programs the focus will be on writing programs.
- They will be marked by the tutorial instructor.
 - The graded quizzes will be returned in subsequent tutorials.
 - After that you can contact him/her for the grade and/or quiz.

Assignments

- They will involve the creation of a working and executable computer program.
- Use a text editor to create it and you will electronically submit the text file for marking.
- Although you may be given some time in tutorial to work on your assignments (during the “open tutorial”) mostly you will complete your work on your own time.
 - Don't underestimate the time/effort required.
 - Creating a good working program is harder than it may first appear.
 - (JT's \$0.02: it's far easier to criticize those crummy 'Apps.' that it is for you to write a good one yourself).
- They will be marked by the tutorial instructor.
 - You can contact him/her for the grade and/or the completed marking sheet.

Submitting Assignments

- **Bottom line: it is each student's responsibility to make sure that the correct version of the program was submitted on time.**
- Late assignments will not be accepted.
- If you are ill then medical documentation is required.
 - Contact your course instructor and not your tutorial instructor with details.



I am the
'course
instructor'
person

- (More on this will be available during the term).

JT's Helpful Hint: Backup Your Assignments

- Sometimes bad things happen!
- One way to mitigate problems is to make copies.
 - Do it often.
 - Do it in many places.
 - Do it as if there is no tomorrow!
- Rules of thumb:
 - Backup important files as often as possible (monthly, weekly or even more often).
 - Backups *should not* be made on the same disk where the data is stored.
 - Use an external hard drive or flash drive that is only connected when the backup is taking place (to mitigate system failures or virus attacks).
Disconnect it otherwise.
 - JT's opinion: this is why I don't like/don't use automated backup programs.

JT's Helpful Hint: Backup Your Assignments (2)

- For truly crucial data consider a second backup that is stored at another location separate from the computer (file attachments via email, CD/DVD's, USB key etc.). Of course you may want to avoid putting private information in your email.
- You can use your Computer Science account as an additional backup location (more on this later).
- After completing the backup check that the backup was successful – backups do periodically fail (reverse the backup process to check that it worked and/or view the contents of the files).

JT's Helpful Hint: Electronically Submitting Work

- Bad things sometimes happen!
 - Sometimes it's a technical failure.
 - Sometimes it's human error.
- Rules of thumb for assignment submissions:
 - Do it early! (Get familiar with the system)
 - Do it often! (If somehow real disaster strikes and you lose everything else at least you will have a partially completed version that your TA can mark).
 - Check your work.
 - Don't assume that everything worked out OK.
 - Instead you should check everything.
 - Don't just check file names but at least take a look at the file contents.

Backing Up And Submitting Your Work

- Bottom line: **it is up to you** to make sure things are done correctly and on time.
- If you have questions beforehand then do ask (make sure you ask your questions early enough so you can receive an answer).
- But don't wait until after the due date (it's too late).

Examinations

- There will be two: midterm and final exam.
- Midterm exam.
 - I set the date, info on course web page:
http://pages.cpsc.ucalgary.ca/~tamj/217/#Assignments_and_exams_
- Final exam.
 - Date/time/location determined by the Office the Registrar.
 - (That means I find out these details at the same time that you do).
 - You can find information about your final exams online via the university PeopleSoft portal.
- Both will completed on paper (not in front of a computer).

Examination Content

- Multiple choice questions:

- Partial program traces e.g., what's the program output
- Basic program structure e.g., find the errors, which function or operator is needed for a particular mathematical operation
- Etc.

- Written questions:

- Write a small/partial computer program.
- Trace the execution of a computer program.
- Conceptual (lower weight for this type of question) e.g., definition of a technical term.
- Likely there will be smaller proportion of written questions on the midterm vs. the final.

- I will be grading these.

- (I'll do the best I can to get them done in a timely fashion but remember it's a high enrollment class).

Examination Content (2)

- More sample 'exam type' questions will be provided during the semester.
 - Sometimes 'on the fly' in lecture so pay attention and take notes.

Estimating Your Term Grade

- As stated in the course information sheet (official signed document) each major component will be awarded a grade point.
 - Individual assignment
 - Midterm exam
 - Final exam
 - Total quiz score
- The mapping of raw score to grade point will be posted before each assignment is due (variation between assignments will occur).
- The mapping of the midterm to grade point will be posted sometime after the midterm.
- The mapping of final to grade point cannot be provided until after the official term marks have been released (Department policy).

Estimating Your Term Grade (2)

- Quiz scoring:

- Each quiz will have a raw score.
- Scores will be normalized so each quiz will be out of 1 mark (equal weight).

Total quiz score	Letter/grade point
5 – 6	A / 4
4	B / 3
3	C+ / 2.3
2	C- / 1.7
1	D / 1.0
0	F / 0.0

Estimating Your Term Grade (3)

- To determine your weighted term grade point simply multiply each grade point by the weight of each component.
- Sum the weighted grade points to determine the term grade.
- Simple and short example (not exactly the same as this term but it should be enough to give you an idea of how to do the specific calculations required this semester):
 - Assignments: weight = 30%, example score = A
 - Midterm: weight = 30%, example score = B+
 - Final: weight = 40%, example score = C-

Weighted assignments: $0.3 * 4.0 = 1.2$

Weighted midterm: $0.3 * 3.3 = 0.99$

Weighted final: $0.4 * 1.7 = 0.68$

Total term grade point = $1.2 + 0.99 + 0.68 = 2.87$

Feedback

What is he talking about???

Wow I am the greatest speaker in the world!



Let me know how things are going in the course:

- Am I covering the material too slowly or too quickly.
- Can you read the slides and my hand writing.
- Can you hear me in the class.
- Etc.

