$Id: syllabus-cmps109.mm,v 1.19 2018-06-19 16:40:47-07 - - $
PWD: /afs/cats.ucsc.edu/courses/cmps109-wm/Syllabus
URL: http://www2.ucsc.edu/courses/cmps109-wm/:/Syllabus/

1. General Information

The generic part of the syllabus contains detailed information about prohibiting cheating, due dates and times, submitting assignments, and verification of the submit. Read it carefully, as you will be held responsible for it.

Directory: The directory `/afs/cats.ucsc.edu/courses/cmps109-wm/` and its subdirectories contain all assignments, handouts, examples, old exams, etc.

Piazza: `https://piazza.com/` is for questions and discussions that are appropriate in the classroom or lab section.

Assignments: All assignments must be submitted electronically and must work on the Unix servers (`unix.ucsc.edu`).

Due Dates: Due dates are announced in the `README` files in the course directory. You must frequently check the `README`. Emailed programs will not be accepted. ***Late submissions will not be accepted except in case of emergency (illness or injury) requiring a physician's attention.***

Cheating: ***Cheating will not be tolerated. See the secion on cheating in the generic part of the syllabus.***

Grades: To pass the course, both the programming component and the testing component will be taken into consideration. Failing either component may be cause to fail the course. Your final grade will be computed as follows:

Programming assignments: 5 × 10% = 50%
Midterm and final tests in class: 2 × 25% = 50%
Summer quarter does not have an exam week.

2. Course Description from Catalog

CMPS-109. Advanced Programming. An introduction to object-oriented techniques of software development including data abstraction, inheritance, polymorphism, and object-oriented design. Extensive practice using a computer to solve problems, including construction of graphical user interfaces and a multithreaded client/server application. **Prerequisites:** CMPS-012B/M or CMPS-013H.

3. Textbooks and References

(1) Bjarne Stroustrup: ***Programming Principles and Practice Using C++***, second edition. Addison-Wesley, 2014. ISBN 0-321-99278-4. This is an elementary textbook for a first course in C++. The previous edition will also work, or any other C++ textbook you may already have.

(2) Avoid any book that does not discuss C++11, and look for a note about C++11 displayed prominently on the cover. Any C++ book not discussing C++11 should be considered obsolete. Prefer books discussing C++14 or C++17.

(3) The C++ Resources Network : Use this site to find specific information about various classes in the standard library.
`http://www.cplusplus.com/`

(4) C++11 : The New ISO C++ Standard FAQ :
`http://www.stroustrup.com/C++11FAQ.html`

(5) Bjarne Stroustrup : ***The C++ Programming Language, 4th edition.*** Addison-Wesley, 2013. This is the revised definitive description of C++11.

(6) Stanley B. Lippman, Josée Lajoie, Barbara E. Moo : ***C++ Primer, 5th edition.*** Addison-Wesley, 2013. A good primer discussing C++11.

(7) Nicolai M. Josuttis : ***The C++ Standard Library, 2nd edition : A Tutorial and Reference.*** Addison-Wesley, 2012. A specific tutorial on the library, with C++11.

(8) David Vandevoorde, Nicolai M. Josuttis, Douglas Gregor : ***C++ Templates : The Complete Guide, 2nd edition.*** Addison-Wesley, 2018.

(9) JTC1/SC22/WG21 — The C++ Standards Committee : Latest publicly available draft : ***N3797 Working Draft, Standard for Programming Language C++.*** 2013-10-13.
`http://www.open-std.org/jtc1/sc22/wg21/`
`http://www.open-std.org/jtc1/sc22/wg21//docs/papers/2013/n3797.pdf`

(10) Bjarne Stroustrup : ***The Design and Evolution of C++.*** Addison-Wesley, 1994. This is a historical document where Stroustrup discusses his design philosophy and how it derived from C with Classes.

(11) P.J. Plauger, Alexander Stepanov, Meng Lee, David Musser : ***The C++ Standard Template Library.*** Prentice-Hall, 2001. Detailed description of the implementation of the STL, showing detailed code examples.

(12) Scott Meyers, `http://www.aristeia.com/books.html`

4. Detailed Syllabus.

This course is about programming in C++, including C++11. Prior programming knowledge of ANSI C is assumed.

(1) C vs C++. Some differences : Input/output, strings, vectors instead of standard I/O, character and other arrays. Fundamental data types. Using the Standard Template Library (STL).

(2) Functions : pass by value, reference, and const reference. Namespaces.

(3) Classes : interface and implementaton. Header files and file guards. Member functions. Operator and function overloading.

(4) Input and output streams. User-defined I/O operators. Formatting I/O.

(5) Vectors and free store. Memory management Constructors and destructors. Copying and assignment of objects. Pointers and references. Shared_ptr and Unique_ptr.

(6) Vectors and arrays. Copy and move constructors, and copy and move assignments. Destructors. Explicit constructors.

(7) Inheritance polymorphism and object-oriented programming. Abstract classes. Virtual functions. Overriding functions and operators. Multiple inheritance as interfaces.

(8) Template polymorphism and generic programming. Containers and inheritance. Range checking and resource management. Exceptions : defining, throwing, and catching.

(9) Containers and iterators in the standard library. Vectors, lists, and strings. Algorithms and maps.

(10) Graphical user interfaces (GUI). Threads. Sockets. Client/server applications.

(11) Miscellaneous other topics : TBA.

5. Students with Disabilities

If you qualify for classroom accommodations because of a disability, please submit your Accommodation Authorization Letter from the Disability Resource Center (DRC) to me as soon as possible, preferably within the first week of the quarter. Contact DRC by phone at 831-459-2089 or by email at `drc@ucsc.edu` for more information.

6. Pair Programming

You may do pair programming if you choose. You are responsible for choosing a partner with whom you can work. Read the guidelines in the directory `pair-programming/`.

7. Submit Checklist

Also read the submit checklist in `submit-checklist/`.

8. Generic Syllabus

`$Id: generic-syllabus.mm.so,v 1.5 2018-01-05 13:17:13-08 - - $`

This is an attachment that I use for the syllabus in all of my courses. Any contradictions between what follows and what is written in the course-specific syllabus is resolved in favor of the course-specific syllabus.

9. Disability Accommodations

It is UCSC policy to assist students with disabilities. If you qualify for accommodations because of a disability, please submit to me your ***Accommodation Authorization Form*** from the Disability Resource Center (DRC) right away. Contact DRC at:

Disability Resource Center	Email: drc@ucsc.edu
146 Hahn Student Services	Phone: (831) 459-2089
1156 High Street	Fax: (831) 459-5064
University of California	
Santa Cruz, CA 95064-1077	

Student responsibilities are as follows:

(1) Students contact the DRC to determine their eligibility for accommodations. When approved by DRC, they will receive their Accommodation Authorization form.

(2) Students then notify their instructor during office hours or after class of their accommodations, and provide their instructor with their Accommodation Authorization Letter. This should be done during the ***first week*** of the quarter.

(3) Please note that it is the student's responsibility to contact the instructor about authorized accommodations. For disability-related testing accommodations, arrangements must be made at least three weeks before each test or exam.

10. Academic Integrity and Cheating

Cheating will not be tolerated. Cheating is defined as giving or receiving unpermitted aid in any programming assignments, examinations, or other course work that is used by the instructor as a basis of assigning grades. Incidents of cheating will be reported to the Provost of the student's college and to the School of Engineering for disciplinary action. Cheating in any part of the course may lead to failing the course and suspension or dismissal from the University.

Be warned that the School of Engineering has a policy of being highly intolerant toward cheating and/or academic dishonesty. All students shall read the UC Santa Cruz Academic Integrity web pages: **`http://www.ucsc.edu/academics/academic_integrity/`**.

Students are expected to maintain high standards of academic honesty. That means that any work submitted by a student which is not completely his/her own is not acceptable. The only exceptions are: code provided by the instructor or TA, whether given in class or provided in the instructor's course directory, and code

taken from the assigned textbook. Specifically, what is not acceptable is swapping code. "Just working together" is not an acceptable excuse. Helping each other with general questions is OK and that is one of the uses of the newsgroup.

Pair programming: For those courses where pair programming is explicitly permitted, it is not cheating for the two partners of a pair to share code, they are developing their projects using a single code base. See the guidelines for pair programming in another directory. It is cheating if the code between different pairs is excessively similar.

Any code not written by the student must be acknowledged in the `README` submitted with the assignment. Submitting code not written personally by the student and which is not acknowledged in the `README` is always cheating, whether or not the code would be otherwise authorized. Submitting code not written personally by the student, even if acknowledged in the `README`, is cheating if it pertains to parts of assignments that the student is expected to write individually. This refers to programs received from anyone, even tutors, or found on the web or other open-source code sources. And if an argument toward honesty does not convince you, note that if you can surf the web to find some code, so can others too, and they may then submit the same code you do.

Getting help from tutors in developing programs is of course expected. Tutors, in the course of their duties, may provide small code fragments, or explain to students how to rewrite their code, or how to debug existing code and modify it in order to get it working.

11. The Final Exam

From the Registrar: ***Final Examination Policies and Schedule.*** Final examinations are given during the exam week period at the time announced in the Schedule of Classes, usually in the same room used for class meetings during the quarter. Final examinations are required in all undergraduate courses. ... If a student misses an examination due to a documented illness or other emergency, the instructor may agree to give an Incomplete and schedule a makeup examination provided that the student's work is passing up to that point. When a final examination is one of the regular requirements in a course, no one taking the course may be individually exempted from it. Travel plans for vacation are not an emergency, and should not be made without checking the final examination schedule.

12. Assignment Specifications and the Final Grade

Late assignments will not be accepted except under extremely unusual circumstances and then only with adequate formal documentation. Make-up tests and exams will not be given, and workarounds for missed tests and exams will be permitted only in case of emergency, such as illness or hospitalization requiring the attendance of a physician.

You can't pass the course without making at least a reasonable attempt at the programming assignments. You can't pass the course without a reasonable performance on the midterm tests and the final exam. This is not an absolute rule, but poor performance on any component of the course will justify a lowering of the general average. For example, someone who gets an '**A**' on the tests and exam and an

'**F**' on the programming projects might get a '**D**' rather than a '**C**' as a final grade. Similarly for the reverse situation. In other words, if there is a significant disparity between the programming results and the testing results, the lower component will have more influence.

A final grade will in general be given by the standard weighted average formula, where the values of $weight_i$ are given under 'Grades' at the beginning of the syllabus:

$$finalscore = \sum_{i=1}^{n}\left(\frac{score_i}{maximum_i} \times weight_i\right) \qquad \textbf{where} \qquad \sum_{i=1}^{n} weight_i = 1.0$$

The `Assignments` subdirectory contains the syllabus and assignments, given in both text and Postscript format. Both are generated from the same `groff` markup source file, but due to the restricted nature of ASCII text, some information may occasionally be missing, such as diagrams or math formulae. An example is the formula above.

If you are reading the ASCII text version of this file, you will see `geqn` markup language, while if you view the Postscript version, you will see the actual equation. The Postscript format version of the assignment is "official", and the only version that has been proofread. In case of discrepancy, refer to the Postscript version of the assignment. The text version is convenient for `grep`ping, but the final word is always in the Postscript version.

13. Due dates and times

A `unix.ucsc.edu` account is required in order to submit assignments. All assignments will be submitted electronically via the Linux timeshares and must work on these machines in order to receive a grade. Submitting an assignment via email is not acceptable.

Graders are required to be logged into the IC Solaris Operating Environment (`unix.ucsc.edu`, etc.), machines when doing any part of the grading. Whether or not an assignment "works" with another operating environment will not influence an assignment score in any way. Students may develop their work with any operating environment they have access to, but must port their programs to IC Solaris before submitting them.

Exactly when are assignments due? Assignments are to be submitted electronically using the submit command any time on or before the date specified as the due date. That includes up to and including 23:59:59 of the specified due date. The time of the lecture is not relevant to the time an assignment is due. In practice, however, you can get away with submitting assignments even later, up until the time that the submit directory is locked with a Unix command. This will done early the following morning at about 09:00:00.

Waiting until the last minute to submit is not a good idea, since after midnight is gremlin time. Gremlins are heavily armed with very sharp pointers. This leaves the exact cutoff time somewhat fuzzy. If you submit the same assignment more than once, the later submission will override the earlier one. ***Submit early! Submit frequently!***

Not knowing how to use the `submit` command is not an excuse for failing to submit

an assignment. It is your responsibility to learn how to use it as well as how to use AFS well in advance of the due date. Once you have done the submit, verify that you have submitted everything that needs to be submitted.

14. Submitting assignments

Firstly, it is important for you to understand how to submit an assignment. Waiting until the due date to find that out will likely mean that you will not be able to submit, and hence will score zero.

Submit homework with the command :

```
submit volume project files ...
```

The volume name is the registrar's catalog code, a hyphen, the instructor's initials, a period, the quarter, and the two-digit year. Using `submit` without operands displays the possible submit volumes. If you specify a submit volume without a project name, the available project names will be displayed.

Finally, in order to verify that your submit is complete, and that the grader will be able to do the build, create a new directory in your personal file space, copy all of the files you have submitted and then perform your own test build.

15. Appeals to scores and inaccuracies in grading

If you disagree with a score assigned to you in a programming project, you should send email to the TA asking for more information and detailing your objections or disagreements. If the number of TAs is not equal to 1, please see further information in the `README`, newsgroup, or `SCORE` report sent to you. There is always a deadline for appealing the scores, which will be given in the email sent to you after grading. For score reports that you receive prior to the last week of classes, this will be seven days after grading is completed. Because of the extremely tight schedule during exam week, a deadline of at most 24 hours will be permitted.

For tests, the time to make note of any grading errors is immediately after the class during which the tests are returned. Test score appeals must be done in person with the test in your possession at that time, not via email.

The instructor will email a gradebook summary to each student as soon as the TA reports the scores to the instructor. If you do not receive such an email, it is your responsibility to contact the instructor and find out why not. The registrar provides the instructor with `@ucsc.edu` usernames as part of the information sent along with the class list, and it is these usernames that are in the gradebook. If that information is missing or inaccurate, you will not receive notification. Since everyone registered for the course is presumed to have a `@ucsc.edu` username, all gradebook related email will be sent to the `@ucsc.edu` username only.

Forwarding your `@ucsc.edu` email to an off-campus location is not recommended. If, for any reason, that address becomes inaccessible, email sent to you will likely bounce. If you do forward your email, it is your responsibility to ensure that the forwarding address remains valid.

16. Organizing your files

Throughout this course, in order to keep your files organized, you should make use of subdirectories. Below are some commands that you should enter, or something similar. You may want to do something similar for other courses and assignments. As with the submit discussion, **`cmps999`** is used as the example course. For other courses, make changes as appropriate.

`cd`
: Change directory to your home directory.

`mkdir private`
: Create a private directory tree that you do not intend anyone else to have access to.

`fs sa private $USER all -clear`
: Set the ACL so that only you have access to the directory.

`fs la private`
: Verify that no one other than you has access to that directory.

`cd private`
: Move down into that directory.

`mkdir cmps999`
: Create a subdirectory for the course cmps999.

`mkdir cmps999/asg1`
: Create a subsubdirectory for your first assignment. Everything having to do with that assignment should be placed in that directory. It is too confusing to have programs from many different assignments in the same directory. This is especially true when using **`Makefile`**s, since each assignment needs to have a separate file, but it must be called **`Makefile`**.