CMSI 185-01
COMPUTER PROGRAMMING

http://myweb.lmu.edu/dondi/fall2014/cmsi185

Fall 2014—Doolan 219, TR 10:50am–12:05pm, 3 semester hours
Office Hours: TR 2–3pm, T 9–10:30am, W 3–5pm,
R 4:30–6pm, or by appointment

John David N. Dionisio, PhD
email: dondi@lmu.edu
Doolan 106; (310) 338-5782

Objectives and Outcomes

This course introduces you to the art and craft of *computer programming*, and in so doing seeks to also expose you to the discipline of computer science and the mindset of computational thinking. Long after you finish this course, my hope is that you will be able to:

1. **State what the field of computing is, and why it is important.**
2. **Craft well-structured, working programs.**
3. **Discern how some ways of programming are better than others.**
4. **Appreciate the value of design and testing in programming.**

In addition to the course-specific content, you are also expected to:

5. **Follow disciplinary best practices throughout the course.**

Prerequisites/Prior Background

No prior technical nor programming background is expected in this course.

Materials and Texts

- John David N. Dionisio and Ray Toal. *Programming with JavaScript: Algorithms and Applications for Desktop and Mobile Browsers*, Jones & Bartlett Learning, 2011.
- Assorted handouts, articles, and sample code to be distributed throughout the semester.

In addition, do not hesitate to look for further information regarding the concepts, techniques, tools, and paradigms that we will discuss.

Course Work and Grading

This course uses standards-based grading: your proficiency in each course objective is directly evaluated according to the outcomes shown on page 3 of this syllabus. Proficiency is measured according to the following key:

+	Advanced proficiency
|	Appropriate proficiency
/	Approaching appropriate proficiency
–	Needs practice and support
O	Not yet evaluated

Your submitted work is used to evaluate these outcomes. Letter grades are then assigned as follows:

	+	|	/	–
A	**many**		*none*	*none*
B		**many**	**few**	*none*
C			**some**	**few**
D				**some**
F				**many**

A–, B+, B–, C+, and C– grades are assigned when there are "close calls" between the above thresholds. Qualitative considerations (e.g., degree of difficulty, effort, class participation, time constraints, overall attitude) may improve proficiency measures. You will receive feedback and proficiency updates after every assignment.

Term Portfolio

Your accumulated work for the semester comprises the *term portfolio*—the final, definitive artifact that demonstrates the proficiencies reached for each course outcome. It is how you show that you have, indeed, accomplished the objectives of this course.

The final proficiency for any given outcome is the statistical mode over the set of assigned proficiencies. Ties are broken through qualitative assessment by the instructor. Incomplete portfolios are evaluated on a case-to-case basis.

Working in Pairs

Pair programming is a software engineering technique where two colleagues work in front of the same device in order to develop software. We will model this approach in the course by having you work in pairs. You must work with a different partner for each assignment, but with the same partner for the entirety of any single assignment. Include both names in each submission and submit one copy per pair. Notify me as soon as possible if you are having trouble partnering up.

Resubmitting Work for Re-evaluation: Once Within Two Weeks of Feedback

Standards-based grading focuses on achieving proficiency, not accumulating scores. Thus, portfolio work may be resubmitted for re-evaluation *once within two weeks of receiving feedback* on it. You may resubmit alone or with the same partner as on the original submission—explicitly indicate which option was chosen. *Please notify me by email* when a resubmission is ready.

You must submit all portfolio work by their respective deadlines—late work detracts from outcome *5b*. An assignment's number is its due date in *mmdd* format. Resubmissions for work whose feedback arrives less than two weeks before the end of the semester must be received by December 12.

Workload Expectations

In line with LMU's *Credit Hour Policy*, the workload expectation for this course is that for every one (1) hour of classroom instruction (50 scheduled minutes), you will complete at least two (2) hours of out-of-class work each week. This is a 3-unit course with 3 hours of instruction per week, so you are expected to complete 3 × 2 = 6 hours of weekly work outside of class.

Attendance

Attendance at all sessions is expected, but not absolutely required. If you must miss class, it is your responsibility to keep up with the course. The last day to add or drop a class without a grade of W is August 29. The withdrawal or credit/no-credit deadline is October 31.

Academic Honesty

Academic dishonesty will be treated as an extremely serious matter, with serious consequences that can range from receiving no credit to expulsion. It is never permissible to turn in work that has been copied from another student or copied from a source (including the Internet) without properly acknowledging the source. It is your responsibility to make sure that your work meets the standard of academic honesty set forth in the *LMU Honor Code and Process.*

Special Accommodations

Students with special needs who require reasonable modifications or special assistance in this course should promptly direct their request to the Disability Support Services (DSS) Office. Any student who currently has a documented disability (ADHD, autism spectrum, learning, physical, or psychiatric) needing academic accommodations should contact DSS (Daum 224, x84216) as early in the semester as possible. All requests and discussions will remain confidential. Please visit *http://www.lmu.edu/dss* for additional information.

Topics and Important Dates

Specifics may change as the course progresses. University dates (italicized) are less likely to change.

August	Introduction to the field of computing (Chapter 1)
August 29	*Last day to add or drop a class without a grade of W*
September	Starting to program; JavaScript specifics; data (Chapters 2 & 3)
October	Statements (Chapter 4)
October 31	*Withdraw/credit/no-credit deadline*
November	Functions (Chapter 5)
November 21	*The Imitation Game* in theaters
November 26–28	*Thanksgiving; no class*
December	Other topics *(time permitting)*
December 12	*Final term portfolios due*

You can view my class calendar and office hour schedule in any iCalendar-savvy client. Its subscription link can be found on the course web site (it's too long to provide in writing).

Tentative Nature of the Syllabus

If necessary, this syllabus and its contents are subject to revision; students are responsible for any changes or modifications distributed in class or posted to the course web site.

Course Outcomes

1	**State what the field of computing is, and why it is important.**	
	For this objective, you are expected to show some knowledge of the field, including but not limited to key concepts, terms, personalities, and events. You should know not only what or who these are but also why they matter. Mainstream news and creative work involving computing can be put in context.	You will work toward this objective by reading/watching assigned material and participating in class discussions. You will show that you have accomplished this by writing answers to assigned questions.
2	**Craft well-structured, working programs.**	
	This objective expects you to write code, and write it well.	You learn to program by…programming. You will be told what you are doing well and what you aren't doing well. Keep doing what you're doing well, and change what you aren't doing well, and you will be fine.
3	**Discern how some ways of programming are better than others.**	
	As programs become more sophisticated, you will find that different approaches will accomplish the same thing. However, some approaches will be better than others.	Class discussion and feedback on your programs will guide you to this objective. Pay attention when you are told that you could have done something in another way.
4	**Appreciate the value of design and testing in programming.**	
	Programming, when not done right, becomes error-prone busy work. Good design and effective testing will keep things in line. Robust testing gives you confidence that new work will not break prior accomplishments.	When you are told to structure something in a certain way, do it, and make sure that you understand why this choice is being made (*Hint:* Ask questions if you don't get it.).
5	**Follow disciplinary best practices throughout the course.**	
5a	*Write code that is easily understood by programmers other than yourself.*	Donald Knuth (Who is he? Look him up.) aptly states that programming is not only telling a computer how to do something, but also telling a person how they would tell a computer to do something.
5b	*Meet all designated deadlines.*	

Sample Standards Development Report

Based on these final proficiencies, the student will get a B. Had the student been more consistent in turning work in on time, *5b* would have been higher and could have easily justified a B+ instead.

1	**State what the field of computing is, and why it is important.**	+
2	**Craft well-structured, working programs.**	+
3	**Discern how some ways of programming are better than others.**	|
4	**Appreciate the value of design and testing in programming.**	|
5	**Follow disciplinary best practices throughout the course.**	
5a	*Write code that is easily understood by programmers other than yourself.*	|
5b	*Meet all designated deadlines.*	/

Totals

+	2
|	3
/	1
–	0
O	0