C S 330 - Concepts of Programng Lang

Winter 2017

Section 001: 130 MARB on M W F from 10:00 am - 10:50 am

Instructor/TA Info

Instructor Information

Name: David Wingate
Office Phone: 801-422-6350
Email: wingated@cs.byu.edu

TA Information

Name: David Hart
Email: davidhart100@gmail.com

Name: Darin Costello
Email: darin.costello@gmail.com

Name: Derek Argueta
Email: darguetap@gmail.com

Course Information

Description

The fundamental tool of a computer scientist and software engineer is the programming language. Programming languages of all shapes and sizes are found under every rock in computing. There are large, mainstream languages that everyone uses regularly. But, there are also small, domain-specific languages that are well-known to those in specific fields but not widely known to those outside.

This course will prepare you to understand the concepts underlying programming languages and to design and implement small ones. Rather than simply surveying a bunch of languages at random, we will focus on specific concepts and languages that best embody those concepts. Even if you don't use these specific languages again, you are likely to encounter the concepts in other languages, and like many things, time spent doing something a different way often causes us to be better users of our regular tools once we return to them.

We will also talk about the basics of implementing interpreters -- and language tools in general -- as well as type checking and automated memory management. Few of us will design a large mainstream language during our careers, but almost all of us will need to design and implement a small language.

Note: This course is undergoing significant revision from previous semesters, so please pardon the construction dust while we work through the changes. The schedule will certainly change as we go, and all assignments should be considered "draft" until the due date for the preceding assignment or discussed in class.

Prerequisites

This class requires CS 240 and its prerequisites (CS 142, CS 224, CS 235, CS 236), each of which we will draw from in this course. You are also expected to already be a competent programmer in multiple languages--in particular, we will assume familiarity with both C++ and Java, which are taught in prerequisite courses.

Learning Outcomes

Programming Language Familiarity
Learn the vocabulary of programming language design, syntax, and semantics
Program Language Flexibility
Be able to write programs using non-imperative language paradigms.

Grading Scale

Grades	Percent

A	93%
A-	90%
B+	87%
B	83%
B-	80%
C+	77%
C	73%
C-	70%
D+	67%
D	63%
D-	60%
E	0%

Grading Policy

Each assignment must be turned in by midnight on the day it is due in order to receive full credit. Late work will be penalized 10% of the assignment maximum score per day, weekends and university holidays excepted. For example, work turned in before the deadline can receive up to 100% credit. Work turned in by midnight of the following business day can receive up to 90%, etc. **Note: The weighting of individual assignments is tentative and likely to change during the semester.**

Teaching Philosophy

The best definition I ever heard of teaching is that it is creating an environment where learning can take place. Notice that the active verbs in that definition are **learning** (your job) and **creating an environment** where that can happen (our job). The learning environment for this class includes the in-class lectures, any assigned readings, homework assignments, programming labs, help sessions with the TA or instructor(s), and even (hopefully) exams. Nothing in the class is busywork. **To get the most out of the class you are encouraged to make use of all of these resources.**

Electronic Readings

Although there is no required textbook, various electronic (mainly online) readings may be assigned throughout the semester.

Software

Instructions for the software you will use during the semester (mainly compilers and interpreters for various languages) will be made as needed throughout the semester.

Coding Your Own Assignments

We strongly encourage you to get together in groups to understand concepts from class, brainstorm ideas, and work through and design algorithms. But **all code that you turn in must represent your own coding effort**. Copying code is not permitted and will be treated as cheating. If you're ever unsure where the line is between collaborating with your fellow students and writing code that reflects your own understanding and ability, please talk with one of the instructors.
If you haven't figured it out already, version control systems can be useful even when working alone. Many of you may already use tools like Github or similar systems to manage your own code for your course projects. If you do so, **make sure any repositories you maintain for your code for this class are private and secure**. The university's Academic Honesty policy defines cheating as not only copying someone else's work but "allowing someone to copy from you while completing an assignment". **If your solution to an assignment for this class is found in an open online repository, it will be considered cheating and treated just the same as if you had knowingly shared your code with someone else.**

Assignments

Assignment Descriptions

Basic Racket

Jan
13

Due: Friday, Jan 13 at 11:59 pm

http://liftothers.org/dokuwiki/doku.php?id=cs330_f2016:racketbasics (http://liftothers.org/dokuwiki/doku.php?id=cs330_f2016:racketbasics)

Here is a basic grader to check your work:solo_grader_basic_rkt.py Download (plugins/Upload/fileDownload.php?fileId=8363a334-00di-9q7Y-oOAN-a219f9bc89e3&pubhash=F9W-uHzJ3-AxU3HJ7NpVRej3Ew16sLvqTSNfX-AWW2ojgjOGseu2K7WCw2bVCtd1Ttcf3nulbRjvNJW-C4wobQ==). This tests basic functionality and the grader we run later this week will be different. (For many future labs, there will a Docker that includes a grader and the correct version of the language.)

Lists and Recursion

Jan
18
Due: Wednesday, Jan 18 at 11:59 pm

http://liftothers.org/dokuwiki/doku.php?id=cs330_f2016:racketlists (http://liftothers.org/dokuwiki/doku.php?id=cs330_f2016:racketlists)
Here is a grader for a few generic tests: solo_grader_lists_rkt.py Download (plugins/Upload/fileDownload.php?fileId=17e75c43-Yhks-Iv0t-wwwG-M0f8df95761c&pubhash=u3kvw3U9REmJIl2ijklTQg7IedZ7RyLTrgKjXS1i70ONmAdszgw5HHt7ZDDd7RLD1p_0ooq4rpXApi-XhUonoQ==).

First-class and higher-order functions

Jan
20
Due: Friday, Jan 20 at 11:59 pm

http://liftothers.org/dokuwiki/doku.php?id=cs330_f2016:rackethof
Submit one racket file containing all your function definitions.
Here is a proto-grader: solo_grader_hof_rkt.py Download (plugins/Upload/fileDownload.php?fileId=9fa200d9-qcFT-5EVB-MVwE-cle6c065b8b2&pubhash=x7-YRN4Nfz0D3vZIwZY1ePPfrIi2CF2MPdnlJLnPSD1FK5zighIEVN4vVU2hHaCzLejlWRGXzTZ4xq63n12Guw==).

More Higher-Order Functions

Jan
25
Due: Wednesday, Jan 25 at 11:59 pm

Submit your deliverables (default-parms, type-parms, and new-sin2) in a racket file.
Here is a proto-grader: solo_grader_hof2_rkt.py Download (plugins/Upload/fileDownload.php?fileId=7e18a387-Te8b-W5qE-eDU0-3e7c2d559759&pubhash=ciDgtXGE81PGLdoEc6c_HF8zp7R0gW6Y5hx7R-LI-A-BZ3FdTfEbCDbKXdlaiXffyTEAAvnTiaa8lont2598BQ==).

Julia Programming 1

Feb
01
Due: Wednesday, Feb 01 at 11:59 pm

Turn in your functions in a Julia (.jl) file.
solo_grader_intro_julia.py Download (plugins/Upload/fileDownload.php?fileId=be4d5bf8-QzhX-AIGI-Xikh-9374d38b4330&pubhash=wf7TDkLakAdQ8LSndUeS9SeizEyLXXn7htR6tMGo8JRbqn_HQFJYRS8g6Rb46DNbdJfodC6F20pjW4hitvjDQA==)

Interpreter 1

Feb
10
Due: Friday, Feb 10 at 11:59 pm

Int1SoloGrader.py Download (plugins/Upload/fileDownload.php?fileId=2a6286ab-CbGC-Kt94-Amln-c66b66c830c4&pubhash=AFYVuXVa6JbemA4cfiAguVRGBn3YRKZ_PkvHcZJi7A4c7w8rmDxE_CK7OUw7L6C6lB0JiqPKHZLvJdptprahUA==)

Interpreter 2

Feb
17
Due: Friday, Feb 17 at 11:59 pm

extint_solo_grader.py.py Download (plugins/Upload/fileDownload.php?fileId=1c8e9fa9-N21H-fIAL-TTpT-Y50645b4dd84&pubhash=rs_-hSkwjlf0M79OXF3ZIh9uIK4A-YajQfyM4shVViJWfrN3ruz-wRivXKhe00jdTmNqI7SaqU-YB82jfIMwUw==)

Interpreter 3

Feb
24
Due: Friday, Feb 24 at 11:59 pm

TransInt_solo_grader.py Download (plugins/Upload/fileDownload.php?fileId=36aa58de-orJI-8htT-x4H8-tUc2a77a4d42&pubhash=CWsTd2PfKeTyru8OZdLZSK9r7mr0QC8puX9BNeWQ_TzKr0PUzuRGHHBf6cs6C7gVmbPMN57eG2xkwTtfQtVxUw==)

Interpreter 4

Mar
03
Due: Friday, Mar 03 at 11:59 pm

We'll grade this lab a bit differently. The'll be three parts. The first will be an auto graded part will syntax and type check . Then we'll run a couple test pictures, including the one the lab specs. And finally an inspection of the code to see if you are multi threading.
The following are things you should check for syntax, There will be a few pass and several fail cases for each new primitive like the following
`(simple_load)` To many or not enough parameters, or not given a string.
`(simple_save)` Given the wrong number of parameters, given a string as a first parameter or given a owl as a second parameter.
'(render_text)` Given wrong number of parameters and checking if your given strings or owls in the correct places....
The rest will follow in the same vain.
You need to make sure your binops, unops and withs work for this lab.
For the picture test, we'll run the example on the lab specs, as well as a couple others and compare the your output to the correct one.

Logic Puzzles

Mar
10
Due: Friday, Mar 10 at 11:59 pm

Prolog 2

Mar
15
Due: Wednesday, Mar 15 at 11:59 pm

Erlang 1

Mar
20
Due: Monday, Mar 20 at 11:59 pm

Erlang 2

Mar
27
Due: Monday, Mar 27 at 11:59 pm

Garbage collection

Mar
31
Due: Friday, Mar 31 at 11:59 pm

Written assignment

Lazy Programming

Apr
07
Due: Friday, Apr 07 at 11:59 pm

Type Checker

Apr
19

Due: Wednesday, Apr 19 at 11:59 pm

given_answer5.txt Download (plugins/Upload/fileDownload.php?fileId=ed53b645-SVKk-dp3w-dzS8-AU052e3ae2fb&pubhash=6pJqtGHBMGsVUlbUfsaRtR3_RsCOtjkhl9EdDT0LFy50UaCKWh48C-yAJaBzP0xntDaAG6iILfHm5sxFlOIKaw==)
given_tests4.txt Download (plugins/Upload/fileDownload.php?fileId=de82a92f-qi9E-ZNT1-ks9o-Tw4e3013fa2d&pubhash=3uIPQJd4sK1hUw6yG6moN4eFn-YpXu_ttJQvjSLAc3vLVoZAEooAYa61RQbsDWcbPM2aE2e1faktZgG_DXF36A==)

Schedule

Date	Topic	Reading	Assignments / Exams
Week 1			
M Jan 09 Monday	Course Introduction		
W Jan 11 Wednesday	Racket: Introduction		
F Jan 13 Friday	Racket: Lists and Natural Recursion		**Basic Racket**
Week 2			
M Jan 16 Monday	**Martin Luther King Jr Day**		
W Jan 18 Wednesday	Racket: First-class and higher-order functions		**Lists and Recursion**
F Jan 20 Friday			**First-class and higher-order functions**
Week 3			
M Jan 23 Monday	Racket: More on higher-order functions		
W Jan 25 Wednesday	-out?-		**More Higher-Order Functions**
F Jan 27 Friday	Introduction to Julia -out?-		
Week 4			
M Jan 30 Monday	General structure of programming language tools		
W Feb 01 Wednesday	Interpreters: parsing, evaluation		**Julia Programming 1**
F Feb 03 Friday	Interpreters: expressions and conditionals		
Week 5			
M Feb 06 Monday	Interpreters: variable binding, substitution, and environments		

W Feb 08 Wednesday	-out?-		
F Feb 10 Friday	Interpreters: functions, scope, and closures		**Interpreter 1**
Week 6			
M Feb 13 Monday	Interpreters: recursion, error handling, parser generators, OO scoping		
W Feb 15 Wednesday			
F Feb 17 Friday	Program analysis		**Interpreter 2**
Week 7			
M Feb 20 Monday	**Presidents Day**		
T Feb 21 Tuesday	**Monday Instruction**		
W Feb 22 Wednesday	Adding high-performance primitives		
F Feb 24 Friday	Parallelizing interpreters		**Interpreter 3**
Week 8			
M Feb 27 Monday	Interpreters: mutation, statements		
W Mar 01 Wednesday			
F Mar 03 Friday	Interpreters: call-by-reference, aliasing, aggregate data structures		**Interpreter 4**
Week 9			
M Mar 06 Monday	Prolog - declarative programming, knowledge base		
W Mar 08 Wednesday			
F Mar 10 Friday	Prolog - models of computation		**Logic Puzzles**
Week 10			
M Mar 13 Monday	Prolog - cut operator		
W Mar 15 Wednesday	Erlang - concurrency		**Prolog 2**
F Mar 17 Friday	**No Classes**		
Week 11			
M Mar 20 Monday	Erlang - let it fail, monitoring		**Erlang 1**
W Mar 22 Wednesday	Continuations		
F Mar 24 Friday	Garbage collection		

Week 12			
M Mar 27 Monday	Garbage collection (cont'd)		**Erlang 2**
W Mar 29 Wednesday			
F Mar 31 Friday	Applicative (eager) vs. normal (lazy) order evaluation		**Garbage collection**
Week 13			
M Apr 03 Monday	Haskell		
W Apr 05 Wednesday			
F Apr 07 Friday	Type systems and checking -out?-		**Lazy Programming**
Week 14			
M Apr 10 Monday	Type systems and checking (cont'd) -out?-		
W Apr 12 Wednesday			
F Apr 14 Friday	Type inference, Polymorphism		
Week 15			
M Apr 17 Monday	"Growing a Language"		
W Apr 19 Wednesday	Last day of university classes (no class for us) Final Exam: 130 MARB 2:30pm - 5:30pm		**Type Checker** **All late work due**
Th Apr 20 Thursday	**Winter Exam Preparation (04/20/2017 - 04/20/2017)**		
F Apr 21 Friday	**First Day of Winter Final Exams (04/21/2017 - 04/26/2017)**		
Week 16			
M Apr 24 Monday			
W Apr 26 Wednesday			