Introduction to Functional Programming: Syllabus

Carlos Encarnación
encarnacion.carlos@gmail.com

November 4, 2014

Contents

1 Introduction

This course is designed to introduce you to functional programming using the programming language Haskell. You will learn that functional programming provides very important features that facilitate reasoning about the code, modularization and provides the glue to combine the resultant components.

2 Prerequisites

We assume that you have some acquaintance with programming, and programming languages. Also, we suppose that you know something about at least one language of the following C++, Java or C#. Moreover, we suppose that you are familiar with elementary algebra.

3 Objectives

3.1 General Objectives

1. Learn to use the basic tools of the Haskell ecosystem (e.g. ghc and ghci).
2. Learn how to conduct equational reasoning.
3. Learn to use mathematical induction to proof properties of functions.

3.2 Specific Objectives

1. Learn to use the GHCi.
2. Learn to use lists to solve programming problems.
3. Learn to use trees to solve programming problems.
4. Learn to derive programs from their specification using equational reasoning.
5. Learn to structure programs in monadic style.

4 Course content

The content of this course have been divided into sections. That said, we proceed to give a brief description of each section.

4.1 Functional programming

In this section we use a couple of examples to illustrate what functional programming is.

4.2 Expressions and types

At this point we explain the concepts of expressions, types and values. In particular, we introduce the concept of type class.

4.3 A plethora of numbers

Here we introduce several type classes related to the concept of number. Moreover, we illustrate the use of numbers in Haskell by constructing some functions over numbers.

4.4 Lists

Now we explain list comprehension and other functions such as *zip* and *zipWith*.

4.5 Let's play Sudoku!

Next, we construct the specification and implementation of a Sudoku solver.

4.6 Mathematical induction

In this part, we introduce mathematical induction over natural numbers and over lists. Also we introduce three important functions *foldr*, *foldl* and *scanl*. We end by solving a very interesting problem called the maximum segment sum.

4.7 Complexity

The topic of complexity is covered in-depth.

4.8 The design of an embedded domain-specific language

This section is devoted to the design of a pretty-printing library.

4.9 Infinite lists

The topic of infinite lists is covered in-depth.

4.10 Imperative functional programming

In addition, we introduce the type class *Monad* together with two important instances such as the *State* monad and the *ST* monad.

4.11 Monadic parsing

In this section we tackle the important problem of parsing using a monadic style.

4.12 A simple theorem prover

Finally, we construct the specification and implementation of a simple theorem prover.

5 Evaluation and Grading Policy

As usual sixty-five points will be graded as exams and thirty-five points will be graded as assignments. In other words, you will have two exams that will be graded as twenty points each on the 5th week and on the 9th week. Finally you will have a final exam on the 11th week which will be graded as twenty-five points. Below you will find a detailed description, see tables 1, 2 and 3. Notice that any assignment delivered after the due date will be grade as half of the original grade.

Table 1: Quizzes

Week	Quiz	Grade (pt)	Scheduled
1	Quiz 1	1	Nov 11th
2	Quiz 2	1	Nov 18th
3	Quiz 3	1	Nov 25th
4	Quiz 4	1	Dec 2nd
5	Quiz 5	1	Dec 9th
6	Quiz 6	1	Dec 16th
7	Quiz 7	1	Jan 6th
8	Quiz 8	2	Jan 13th
9	Quiz 9	2	Jan 20th
	Total	11	

Table 2: Homeworks

Week	Assignment	Grade (pt)	Scheduled	Due date	Deadline
1	Homework 1	3	Nov 6th	Nov 18th	Nov 20th
2	Homework 2	3	Nov 13th	Dec 25th	Dec 27th
3	Homework 3	3	Nov 20th	Dec 2nd	Dec 4th
4	Homework 4	3	Nov 27th	Dec 9th	Dec 11th
5	Homework 5	3	Dec 4th	Dec 16th	Dec 18th
6	Homework 6	3	Dev 11th	Jan 6th	Jan 8th
7	Homework 7	3	Dec 18th	Jan 13th	Jan 15th
9	Homework 8	3	Jan 8th	Jan 20th	Jan 22th
	Total	24			

Table 3: Exams

Week	Exam	Grade (pt)	Scheduled
4	Midterm Exam	20	Nov 27th
8	Midterm Exam	20	Dec 23th
11	Final Exam	25	Jan 15th
	Total	65	

6 Calendar

Tuesday		Thursday	
$\boxed{\text{Nov 4th}}$ Lecture 1: § 4.1 and § 4.2	**1**	6th Lecture 2: § 4.3 and § 4.4 Assign Homework 1	**2**
11th Quiz 1 Lecture 3: § 4.5	**3**	13th Lecture 4: § 4.5 Assign Homework 2	**4**
18th Deliver Homework 1 & Quiz 2 Lecture 5: § 4.6	**5**	20th Lecture 6: § 4.6 Assign Homework 3	**6**
25th Deliver Homework 2 & Quiz 3 Lecture 7: § 4.7	**7**	27th First Midterm Exam Assign Homework 4	**8**
$\boxed{\text{Dec 2nd}}$ Deliver Homework 3 & Quiz 4 Lecture 8: § 4.7	**9**	4th Assign Homework 5 Lecture 9: § 4.8	**10**
9th Deliver Homework 4 & Quiz 5 Lecture 10: § 4.8	**11**	11th Lecture 11: § 4.9 Assign Homework 6	**12**
16th Deliver Homework 5 & Quiz 6 Lecture 12: § 4.9	**13**	18th Assign Homework 7 Lecture 13: § 4.10	**14**
23rd Second Midterm Exam	**15**	25th No class - Vacation	
30th No class - Vacation		$\boxed{\text{Jan 1st}}$ No class - Vacation	
6th Deliver Homework 6 & Quiz 7 Lecture 14: § 4.10	**16**	8th Assign Homework 8 Lecture 15: § 4.11	**17**
13th Deliver Homework 7 & Quiz 8 Lecture 16: § 4.11	**18**	15th Lecture 17: § 4.12	**19**
20th Deliver Homework 8 & Quiz 9 Lecture 18: § 4.12	**20**	22nd Final Exam	**21**

7 Literature

1. *Thinking Functionally with Haskell.* Richard Bird. Cambridge University Press, 2015.

2. *Haskell 2010 Language Report.* Simon Marlow (editor). 2010.

3. *Real World Haskell.* Bryan O'Sullivan, Don Stewart, and John Goerzen. 2008.

4. *Programming in Haskell.* Graham Hutton. Cambridge University Press, 2007.

7.1 Suggested reading

1. https://www.fpcomplete.com/school