CS3723 Homework #2, Due: 9AM, 9/10/10

Scala (Chapter 1 in Odersky)

1. In Scala “type annotations” are optional, in what way does this influence the syntax (the rules for how things are written textually) for variable and function declarations?

Computability (Chapter 2 in Mitchell)

2. Why are programs best considered to represent partial rather than total functions?

3. For each of the following function definitions over the domain of integers in the interval \([-10,10]\), list the sets of pairs that the function represents. Indicate whether the function is a partial or a total function over integers in the interval \([-10,10]\).

   (a) \( f(x) = \text{if } x + 3 > 2 \text{ then } x \times 5 \text{ else } 1 / x \)

   (b) \( f(x) = \text{if } x < -2 \text{ then } 3 \text{ else } f(x-2) \)

   (c) \( f(x) = \text{if } x = 0 \text{ then } 1 \text{ else } f(x-2) \)

   [Adapted from Mitchell, Exercise 2.1]

4. List two ways of formally defining the class of computable functions?

5. What does it mean to say that a language is Turing-complete?

6. List three computer languages (including one not mentioned in class) that are Turing-complete?

7. Name one language (not mentioned in class) that you think is not Turing-Complete? What is it about the language that makes you think it is not Turing-Complete? If you use any web sites or other references to figure this out, please cite them.

8. * On one of the other department Linux machines (e.g., main211.cs.utsa.edu) look at the man page for \texttt{sh}. Do you think \texttt{sh} is Turing-complete? Why or why not?

9. What does the Church-Turing thesis tell us about the relative power (in terms of the kinds of functions that are computable) of the various Turing-complete programming languages?

10. Usually, most general-purpose programming languages are considered to be Turing-Complete; this is a slight simplification. What limitation(s) does a real program in a real programming language running on a real computer have that would make it impossible to successfully executing an algorithm that solves a problem that is, in principle, computable?

11. * Assuming you are given a program \texttt{Halt} that can be used to determine whether a program that requires no input halts, is it possible to solve the Halting Problem using \texttt{Halt}0? If so, explain your answer by describing how a program solving the halting problem would work. If you do not think \texttt{Halt}0 would allow one to solve the Halting Problem, explain briefly why you think so? [Adapted from Mitchell, Exercise 2.2]

Compilation and Interpretation (Section 4.1, Lecture Notes #3)

12. What is the name of the process by which a program written in some programming language is translated into instructions for some hardware machine?
13. What kind of program executes another program ‘P’ written in some programming language ‘L’ by simulating ‘L’ and *directly performing* the tasks required by the statements in ‘P’ as it executes those statements?

14. What advantages does compilation have over interpretation? What advantages does interpretation have over compilation?

15. Which is usually faster, the interpretation of some program directly or the direct execution of machine code compiled from it?

*Note: If you collaborated with your classmates or used their notes, please note which classmates you collaborated with. If you use an external source, besides the text book, lectures, notes provided by the instructor, and your own intellect, please cite that source. Use quote marks if you are quoting material word-for-word from any source (including the text book).*