Quiz 1 (day 4)

September 10, 2013

Get out two sheets of paper (or tear one in half). Put your name and student ID clearly on each one. Write "Q1", "Q2", "Q3", "Q4", "Q5", and "Q6" on each sheet. For each question, write your answer on *both* sheets in the appropriate place. Hand in one copy.

In every case, select the single *best* answer.

You're done with any question you answer correctly on the quiz.

By Tue 8PM, submit corrections. Include your name, student ID, the quiz number, and your collaboration statement. (**Even if you got everything right, please do submit for our records**. Just indicate that you got everything correct.) For each question you got incorrect, write the question number and then explain why the correct answer is correct and why the answer you chose is incorrect.

Question 1: Desugaring

Which of these is an example of desugaring?

1. Writing a new, more efficient version of a particular Racket program that uses vectors wherever the old version used lists.
2. Turning input like `7 + 2 * 3` into a value like `(addC (numC 7) (multC (numC 2) (numC 3)))`.
3. As a first step in compilation, replacing `for(A; B; C) D` with the equivalent code `{A; while (B) { {C} {D} }}`.
4. Proving that anywhere an expression like `X + Y` appears in a language, it's means exactly the same as `Y + X`.

•

Question 2: Static vs. Dynamic

In a language that's *statically* typed, you apply the `song-lyrics` function—which expects a `song` value—to a number. When would you expect to find out about the error?

1. Before the program is run.
2. When the program is run, regardless of whether incorrect code is ever evaluated.
3. When code that only works on a `song` is passed a number.
4. When incorrect output guides our debugging to the error.

•

Question 3: Static vs. Dynamic

In a language that's *dynamically* typed, you apply the `song-lyrics` function—which expects a `song` value—to a number. When would you expect to find out about the error?

1. Before the program is run.
2. When the program is run, regardless of whether incorrect code is ever evaluated.
3. When code that only works on a `song` is passed a number.
4. When incorrect output guides our debugging to the error.

-

Question 4: Parsers

We create a new language called "Clarity" based on the language "Perl". Which of these might we expect a parser for the Clarity programming language to do?

1. Convert a Perl program into an equivalent Clarity program.
2. Convert a Clarity program from text to an internal representation.
3. Run a Clarity program and present us with its output and final value.
4. Determine whether a Clarity program will run to completion

-

Question 5: Semantics

Which of these illustrates a difference in *semantics*?

1. Racket can evaluate expressions from the "read-eval-print" loop or run programs from the command line.
2. In JavaScript, `+` means addition when applied to numbers but concatenation when applied to strings.
3. C++ code can accomplish the same tasks with `if/else` statements or `switch` statements.
4. Anonymous functions in Python start with `lambda` but in C++11 they appear in square brackets `[...]`.

•

Question 6: plai-typed

Consider the following `plai-typed` function:

```
; Given a list of numbers, produces the length of the list
(define (length [lon : (listof number)]) : number
  (cond [(empty? lon) 0]
        [(cons?  lon) (+ 1 (length (rest lon)))]
        [else         (error 'length "not a list!")]))
```

In `plai-typed`, `cons?` tests whether a value is a non-empty list.
Why will the `else` case *never* be evaluated?

1. The recursive call in the `cons?` case keeps the function from ever reaching the `else` case.
2. A program that calls `length` on something other than a list will fail before it even begins to run.
3. A function's designer should assume that callers follow the function's contract.
4. In `plai-typed`, every value must evaluate to true when given to either `empty?` or `cons?`.