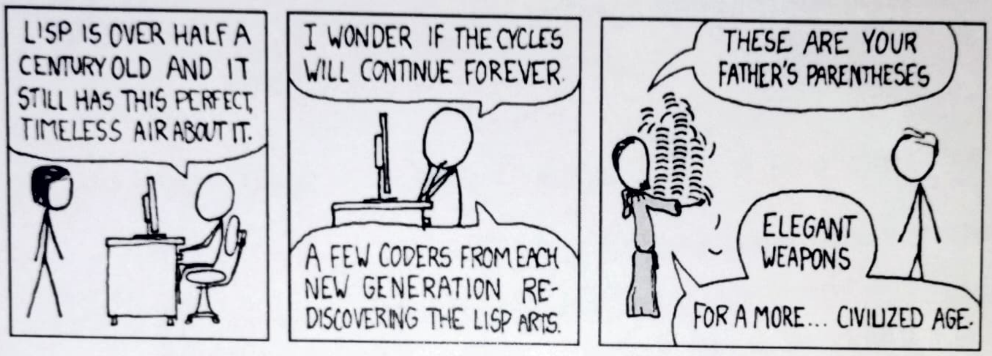
CS339: Abstractions and Paradigms for Programming

Quiz 1 (55 minutes; 10 marks)

August 26th, 2024

Instructions:

- Write neat, clear and crisp answers.
- In case you make an assumption, write it down before the corresponding answer.


LISP IS OVER HALF A CENTURY OLD AND IT STILL HAS THIS PERFECT, TIMELESS AIR ABOUT IT.
I WONDER IF THE CYCLES WILL CONTINUE FOREVER.
A FEW CODERS FROM EACH NEW GENERATION RE-DISCOVERING THE LISP ARTS.
THESE ARE YOUR FATHER'S PARENTHESES
ELEGANT WEAPONS
FOR A MORE... CIVILIZED AGE.

Q1 [2] Determine and explain the results of evaluating the following Scheme expressions:

A. `(null?  (cddr (list (list 2 3 4) 5)))`

B. `(((lambda (x y) (lambda (z w) (* (+ z y) (* x w)))) 3 5) 2 7)`

Q2 [3] A. Assuming the zeroth and the first fibonacci numbers to be 0 and 1, respectively, define a procedure `(fib n)` that returns the nth fibonacci number, such that the generated process is iterative. Show the generated process's behaviour for the application `(fib 5)`. [2]

B. Explain tail-call optimization with the process generated in part A. [1]

Q3 [3] Say we want to enrich the lambda calculus by adding support for pairs. A clever definition of `cons` and `car` is given below:

- `cons = λf.λs.λb. b f s`
- `car = λp. p true`

where `true` works as usual: `true x y` returns `x`.

A. Show that `car` works; i.e., `car (cons v w)` returns v. [2]

B. Give a definition of `cdr` that works too. [1]

Q4 [2] Assume we have a procedure append that, given two lists, appends the second at the end of the first:

```
> (define l1 (list 2 3 4))
> (define l2 (list 5 6))
> (append l1 l2)
(2 3 4 5 6)
```

Use append to define a procedure `reverse` that reverses a list. That is:

```
> (reverse (list 2 3 4 5 6))
(6 5 4 3 2)
```

[Bonus; 1 PC] Can you define `reverse` using the higher-order function `foldl`?