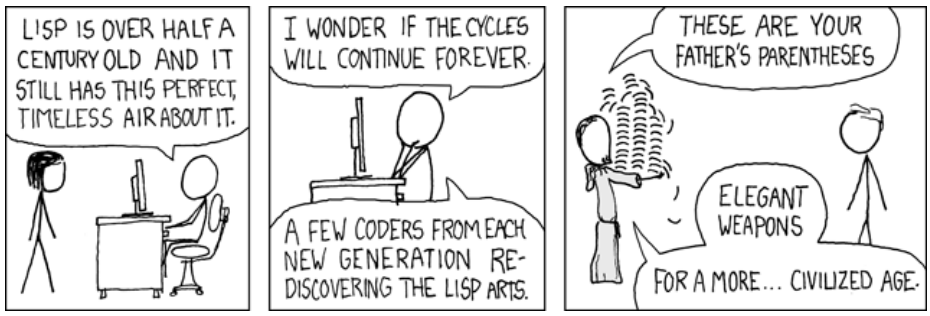
CS302: Paradigms of Programming

Open Book Test (20 points)

August 29^{th}-30^{th}, 2020 (IIT Mandi)

Instructions:

- Submit all the answers in a single file, named *rollnum-obt.pdf* or *rollnum-obt.txt*, on Moodle.
- Each point corresponds to 0.25 marks in the course.
- *Plagiarism policy.* You are supposed to attempt this paper individually. No discussions allowed. If I suspect plagiarism, there will be a spontaneous telephonic Viva (over the whole syllabus) and marks would be awarded only upon satisfactory performance. Avoid trouble – be honest with yourself.



Source: `https://imgs.xkcd.com/comics/lisp_cycles.png`

Q1 [4; Week 9] Determine and explain (both needed to fetch points) the outputs of the programs `P1`, `P2`, `P3` and `P4` that use continuations:

```
; P1
(+ 4 (call/cc (lambda (cont) 13)))

; P2
(+ 4 (call/cc (lambda (cont) (cont 10) 13)))

; P3
(define foo #f)
(+ 4 (call/cc (lambda (cont) (set! foo cont) 13)))

; P4 (in sequence with P3)
(foo 100)
(foo 416)
```

Q2 [5; Week 3] For the following fibonacci function:

$$F = \lambda f.\lambda n.\ if\ (<\ n\ 2)\ n\ (+\ (f\ (-\ n\ 1))\ (f\ (-\ n\ 2)))$$

show that $((Y\ F)\ n)$ correctly computes fibonacci(n), where Y is the Y-combinator. You can use backslashes ('\') to depict lambdas, unless you are using LaTeX.

Q3 [6; Week 13] Say we define an ADT in Haskell to construct lists by adding elements to the end (instead of front):

```
data List a = Nil | Snoc (List a) a
```

Note that `Snoc` here is like a backward `cons`. Thus, the list `[1,2,3]` in this view would be represented as `Snoc (Snoc (Snoc Nil 1) 2) 3`.

First define two functions `scar` and `scdr` that work like `car` and `cdr` for the snoc-view of lists [1+1], and then define two more functions for converting one view of lists to the other [2+2]:

```
toList :: [a] -> List a
fromList :: List a -> [a]
```

Q4 [4; Week 12] Recall that A4 suffered from *order-independence* of pure logic, thus resulting into imprecise results with respect to the control-flow of the input programs. For example, the object `o1` in the following code escaped (along with `o2`):

```
alloc(a,o1)
alloc(a,o2)
invoke(m,[a])
```

Discuss how would you approach the problem to impart *flow-sensitivity*. As an instance, we want to be able to identify that only `o2` escapes in the program that would have generated the above facts. (This question doesn't ask you to write the whole Prolog solution.)

Q5 [1; Free ~~variable~~question] The course-summary blog discussed some of the challenges with online teaching/learning. Now that we have to live in this mode for much longer than what was anticipated in March, the biggest challenge I believe is *ensuring the quality of your BTech degree doesn't reduce much, while also factoring in the difficulties involved.* We all tried whatever we could through the 7-month period; however, feedback helps plan for the future. So please fill the short anonymous course feedback at `https://bit.ly/cs302-spring20`. (The purpose of this is a bit different from the institute TCF; please fill that separately, when it's out.) Finally, write "Done." as the answer for Q5 **after** you have filled the feedback.

--*-*- See you (hopefully literally) in the end-sem -*-*-*-*