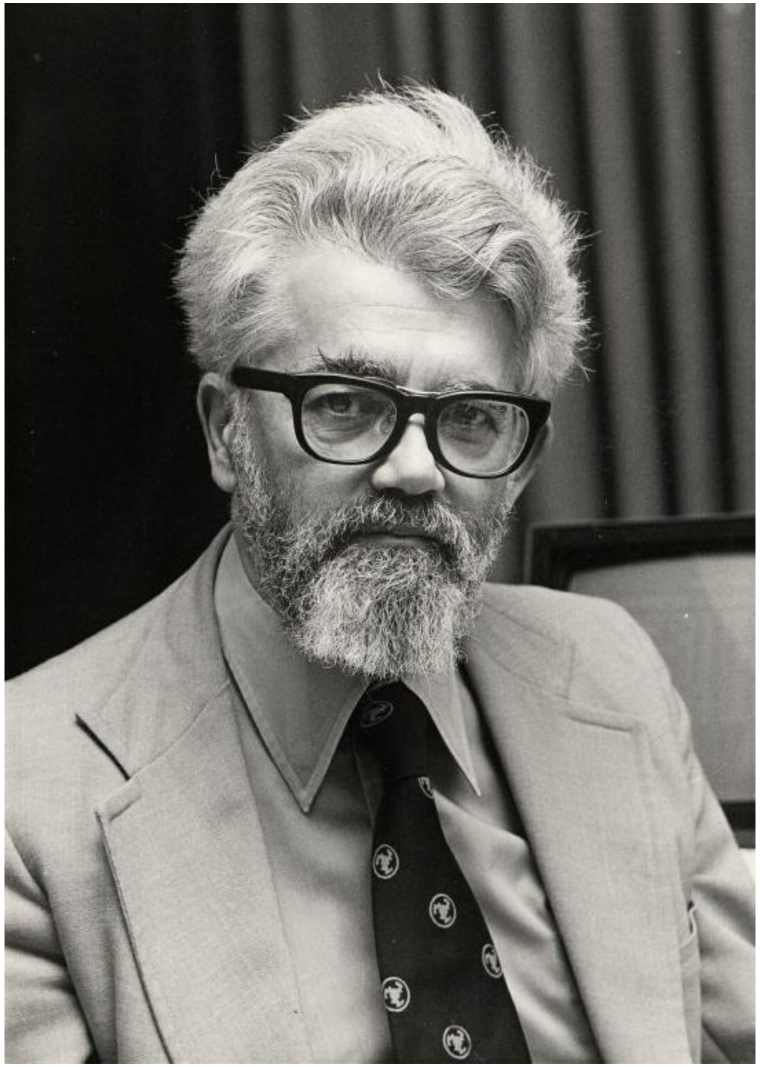
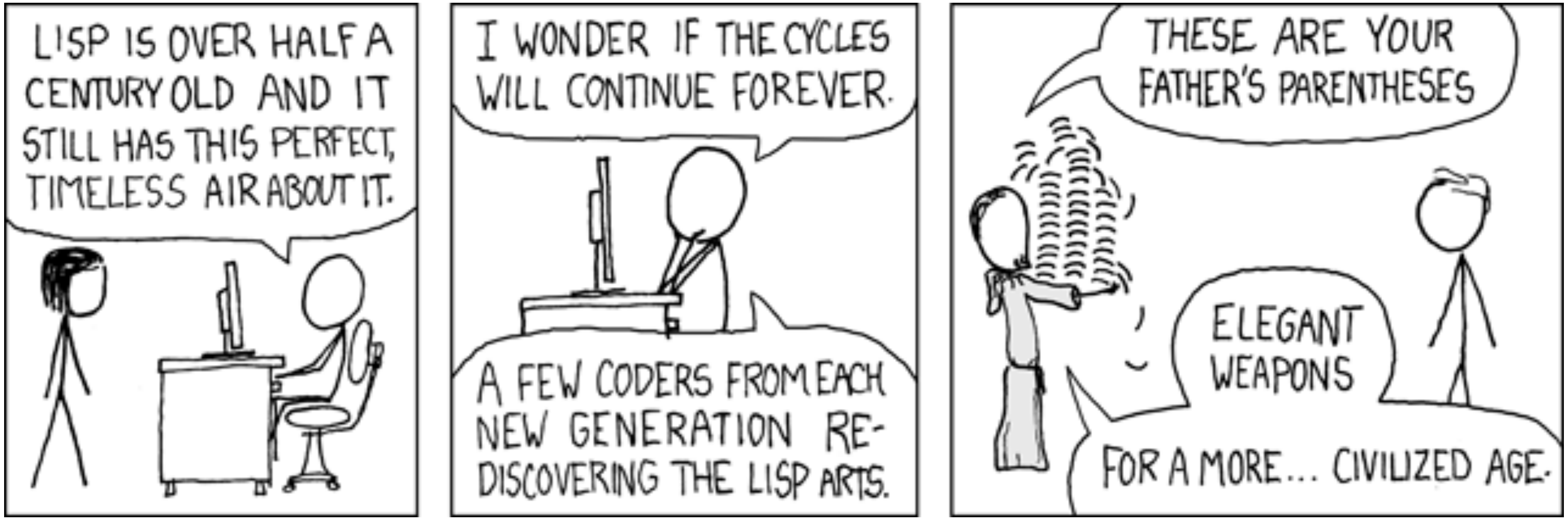
A quick introduction to Lisp

Vish Singh
vishvajitsingh@gmail.com
http://www.lisptoronto.org/

Lisp was invented by John McCarthy..



in 1958.


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.

http://xkcd.com/297/

Lisp syntax

```
(defun factorial (n)
  (if (zerop n)
    1
    (* n (factorial (1- n)))))
```

Lisp stands for List Processing

Lisp code consists of lists of symbols and other objects.

It's easy for Lisp code to manipulate Lisp code.

What can you do with this?

- Teach the compiler new optimizations
- Write new operators
- Create domain-specific languages
    - prolog
    - infix
    - loop

What's modern Lisp like?

- many data structures:
  - linked lists, arrays, trees, hashtables, structs, classes

- many paradigms:
  - imperative, functional, OO, LOP

- usually compiled, not interpreted
  - compiler-macros
  - type annotations

Some modern Lisps

- Common Lisp
  - *CMUCL, SBCL, CLisp, LispWorks, Allegro CL, etc.*

- Scheme
  - *Racket, Chicken, Bigloo, MIT Scheme, etc.*

- Clojure

- Arc

```
(import
 '(java.awt Color Graphics Dimension)
 '(javax.swing JPanel JFrame))

(use 'clojure.contrib.complex-numbers
     'clojure.contrib.generic.arithmetic
     'clojure.contrib.generic.math-functions)

(defn make-panel [w h render]
  (doto (proxy [JPanel] []
          (paint [#^Graphics g] (render g)))
    (.setPreferredSize (Dimension. w h))))

(defn make-frame [& panel-args]
  (doto (JFrame.)
    (.add (apply make-panel panel-args))
    .pack
    .show))
```

```
; Returns the number of iterations for |z| to exceed 2
; or nil if it never does.
(defn num-mandelbrot-iterations [#^complex c max-iter]
  (loop [z (complex 0.0 0.0) num-iter 0]
    (if (> num-iter max-iter)
      nil
      (if (> (abs z) 2.0)
        num-iter
        (recur (+ (* z z) c) (inc num-iter))))))
```

```clojure
(defn mandelbrot []
  (make-frame 768 512
              (fn [#^Graphics g]
                (doseq [x (range 768) y (range 512)]
                  (let [num-iter (num-mandelbrot-iterations
                                  (complex (+ (/ x 256.0) -2.0)
                                           (+ (/ y 256.0) -1.0))
                                  30)]
                    (.setColor g (Color. (if (nil? num-iter)
                                           0
                                           (* num-iter 8))
                                         0
                                         0))
                    (.fillRect g x y 1 1))))))
```