#lang bit-matching

(bitwise-and Fred_Fu Joshua_Larkin)

Grammar

A bit-matching pattern (BMP) is a

<(x n) …> where x is the bit-value of a bit-string of length n

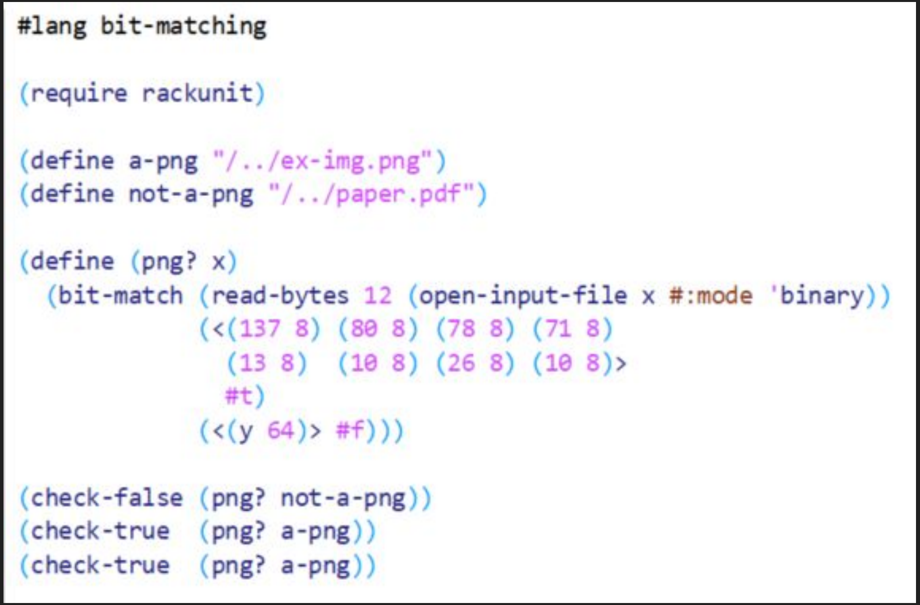
```
(bit-match Expr1
   (BMP Expr2) ...)
```

When the Expr1 is equivalent to the bit string described by the BMP, the result is Expr2.

How it went

- We went through a long period of trying to get `brag` to work with our grammar/syntax and eventually abandoned that, instead using an extension to Racket’s readtable
- The project was a big exercise in deleting code, which was pretty fun
- We added a pattern language to racket, under the domain `bit-matching` so programmers have an easier way to code with binary strings. As an example, we can now write a predicate that determines if a file is of the extension .png

Examples of deconstruction/construction



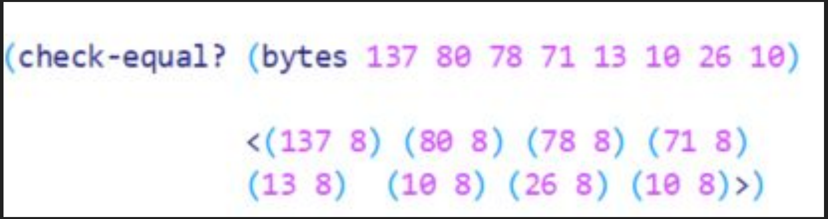
```
#lang bit-matching

(require rackunit)

(define a-png "/../ex-img.png")
(define not-a-png "/../paper.pdf")

(define (png? x)
  (bit-match (read-bytes 12 (open-input-file x #:mode 'binary))
             (<(137 8) (80 8) (78 8) (71 8)
               (13 8)  (10 8) (26 8) (10 8)>
               #t)
             (<(y 64)> #f)))

(check-false (png? not-a-png))
(check-true  (png? a-png))
(check-true  (png? a-png))
```



```
(check-equal? (bytes 137 80 78 71 13 10 26 10)

              <(137 8) (80 8) (78 8) (71 8)
              (13 8)  (10 8) (26 8) (10 8)>)
```