Lab 3
SYSC 3101A
L3E
Nathan MacDiarmid
101098993

```
#lang racket

; EXERCISE 1

(define (build-list n f)
  (cond
    [(= n 0) `()]
    [(cons (f n) (build-list (- n 1) f))]
    )
  )

; build-naturals

(define (build-naturals n)
  (reverse (build-list n (lambda (x) (- x 1))))
  )

; build-rationals

(define (build-rationals n)
  (reverse (build-list n (lambda (x) (/ 1 x))))
  )

; build-evens

(define (build-evens n)
  (reverse (build-list n (lambda (x) (* (- x 1) 2))))
  )

; EXERCISE 2

(define (cubic a b c)
  (lambda (x)
    (+ (+ (* (* x x) x) (* (* x x) a)) (+ (* b x) c))
    )
  )
```

```
; EXERCISE 3

(define (square x) (* x x))
(define (inc x) (+ x 1))

(define (twice f)
  (lambda (x)
    (f (f x))
    )
   )
```