

```
1 #lang racket
2 5
3
4 (lambda () (printf "Hello world!"))
5
6 (define hello-world (lambda () (printf "Hello
6 world!")))
7
8 (hello-world)
9
10 ((lambda () (printf "Hello world!")) )
11
12 5
13
14 (define x 5)
15
16
17 (define sum-squares (lambda (lst)
18                       (if (= 1 (length lst))
19                           (* (first lst) (first lst))
20                           (+ (* (first lst) (first
20 lst))
21                               (sum-squares (rest
21 lst))))))
22
23 (define numbers (list 1 2 3))
24
25 (define (square x)
26   (* x x))
27
28 (define sum-squares-2 (lambda (lst)
29                       (if (= 1 (length lst))
30                           (square (first lst))
31                           (+ (square (first lst))
32                               (sum-squares-2 (rest
32 lst))))))
33
34 (sum-squares-2 numbers)
35
```

```

36 (define (sum-squares-3 lst)
37   (let ((square-1 (lambda (x) (* x x))))
38     (if (= 1 (length lst))
39         (square-1 (first lst))
40         (+ (square-1 (first lst))
41            (sum-squares-3 (rest lst)))))
42
43 (sum-squares-3 numbers)
44
45 (define (reverse-helper str n)
46   (if (= n (string-length str))
47       ""
48       (string-append (reverse-helper str (+ n 1))
49                      (string (string-ref str n)))))
50
51 (reverse-helper "cat" 0)
52
53 (define (reverse str)
54   (reverse-helper str 0))
55
56 (define (reverse-2 str)
57   (letrec ((helper (lambda (str x) (if (= x
57 (string-length str))
58                                     ""
59                                     (string-append
59 (helper str (+ x 1))
60 (string (string-ref str x)))))))
61   (helper str 0))
62
63 (reverse-2 "cat")
64

```