```
#lang racket

(define outfile (open-output-file #:exists 'truncate "class.txt"))

(write "cat" outfile)

(write (quote (string-append "cat" "!")) outfile)

(close-output-port outfile)

(define infile (open-input-file "class.txt"))

(read infile)

(define input (read infile))

input

'input

'(1 2 3)

;You can call (eval) to evaluate the result,
 but only in the REPL (for reasons we'll learn later).

 (eval input)


(write #f)

#f

(print #f)

(printf "\n")

(write (quote (lambda (x)(+ x 5))))

(printf "\n")

(print (quote (lambda (x)(+ x 5))))

(printf "\n")

(write (list 1 2 3))

(printf "\n")

(print (list 1 2 3))

'(1 2 3)
```

```
48
49 (define five 5)
50
```