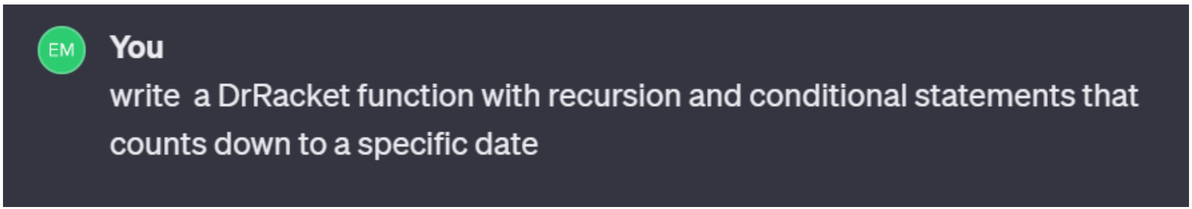
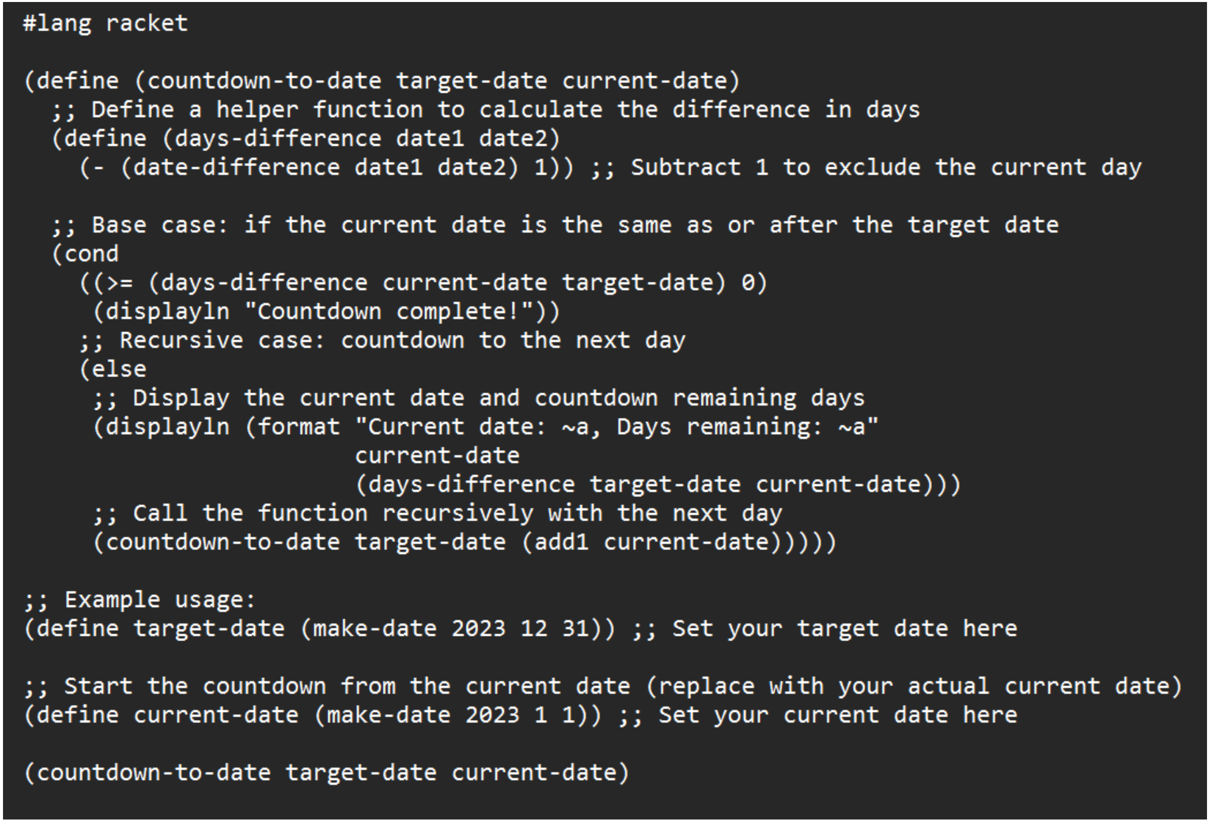
Screenshots of Interactions with ChatGPT

Prompt 1



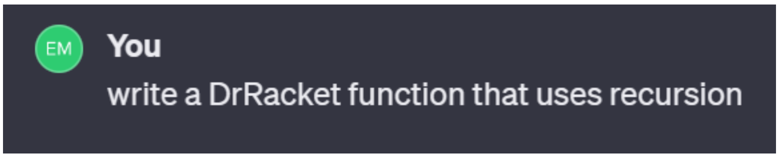
You

write a DrRacket function with recursion and conditional statements that counts down to a specific date



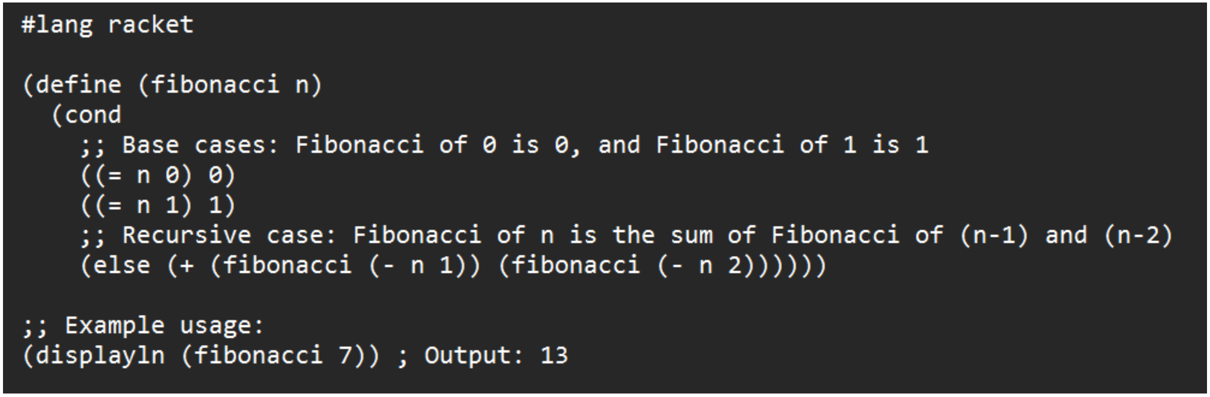
```
#lang racket

(define (countdown-to-date target-date current-date)
  ;; Define a helper function to calculate the difference in days
  (define (days-difference date1 date2)
    (- (date-difference date1 date2) 1)) ;; Subtract 1 to exclude the current day

  ;; Base case: if the current date is the same as or after the target date
  (cond
    ((>= (days-difference current-date target-date) 0)
     (displayln "Countdown complete!"))
    ;; Recursive case: countdown to the next day
    (else
     ;; Display the current date and countdown remaining days
     (displayln (format "Current date: ~a, Days remaining: ~a"
                        current-date
                        (days-difference target-date current-date)))
     ;; Call the function recursively with the next day
     (countdown-to-date target-date (add1 current-date)))))

;; Example usage:
(define target-date (make-date 2023 12 31)) ;; Set your target date here

;; Start the countdown from the current date (replace with your actual current date)
(define current-date (make-date 2023 1 1)) ;; Set your current date here

(countdown-to-date target-date current-date)
```

Prompt 2

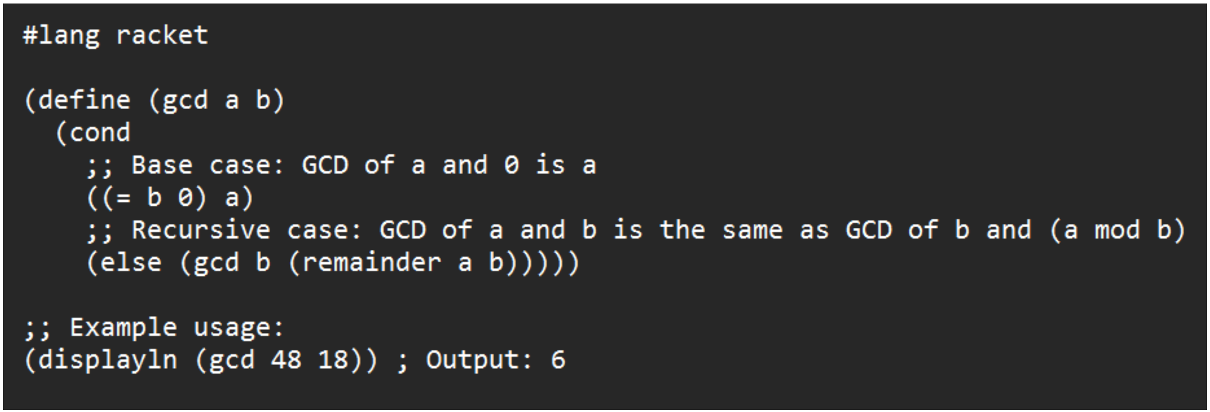


You

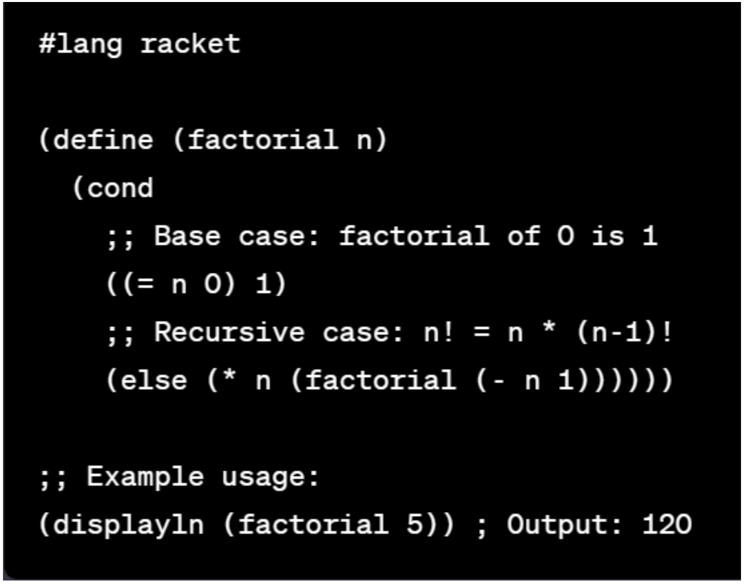
write a DrRacket function that uses recursion



```racket
#lang racket

(define (fibonacci n)
  (cond
    ;; Base cases: Fibonacci of 0 is 0, and Fibonacci of 1 is 1
    ((= n 0) 0)
    ((= n 1) 1)
    ;; Recursive case: Fibonacci of n is the sum of Fibonacci of (n-1) and (n-2)
    (else (+ (fibonacci (- n 1)) (fibonacci (- n 2))))))

;; Example usage:
(displayln (fibonacci 7)) ; Output: 13
```

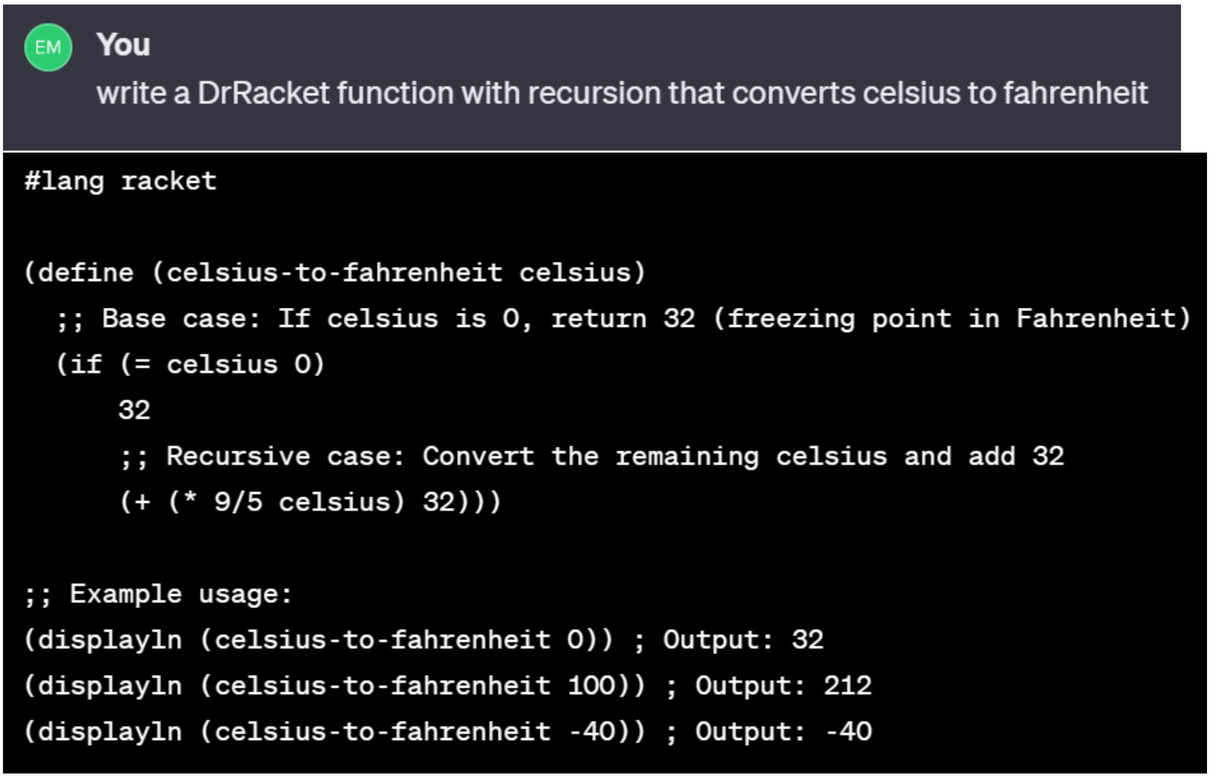


```racket
#lang racket

(define (gcd a b)
  (cond
    ;; Base case: GCD of a and 0 is a
    ((= b 0) a)
    ;; Recursive case: GCD of a and b is the same as GCD of b and (a mod b)
    (else (gcd b (remainder a b)))))

;; Example usage:
(displayln (gcd 48 18)) ; Output: 6
```



```racket
#lang racket

(define (factorial n)
  (cond
    ;; Base case: factorial of 0 is 1
    ((= n 0) 1)
    ;; Recursive case: n! = n * (n-1)!
    (else (* n (factorial (- n 1))))))

;; Example usage:
(displayln (factorial 5)) ; Output: 120
```

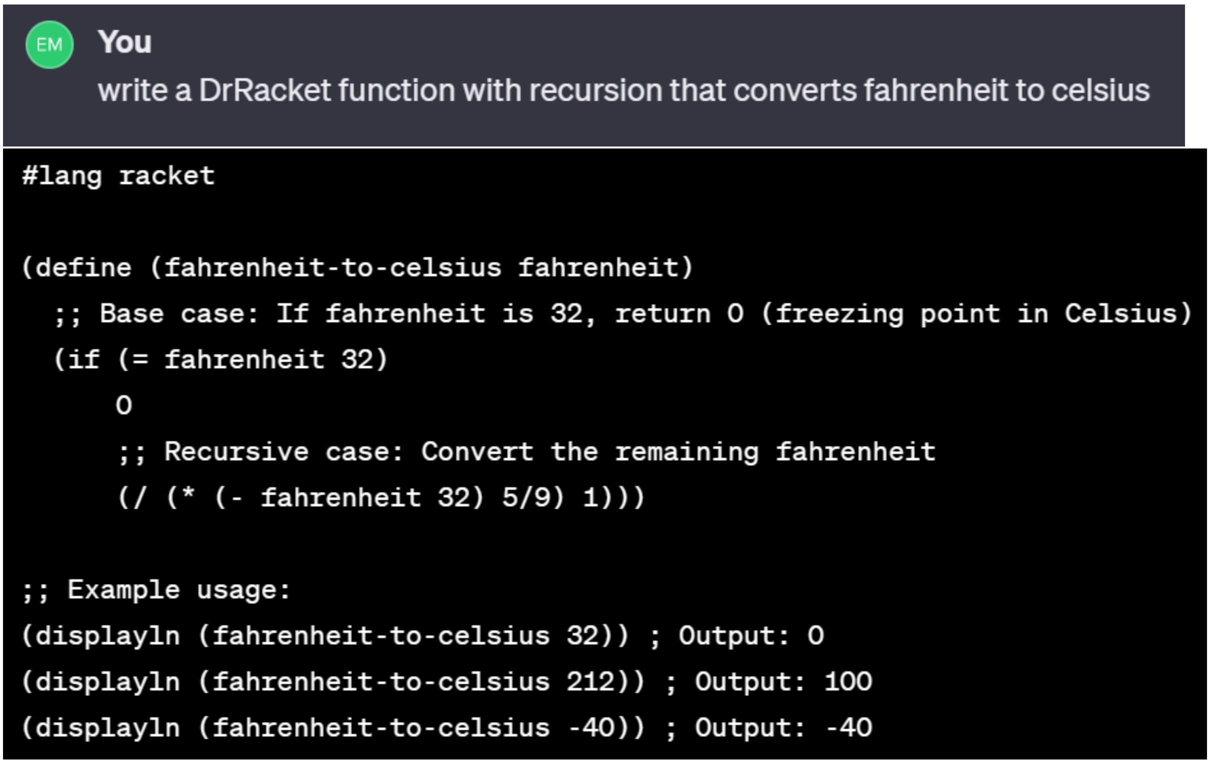
Prompt 3



write a DrRacket function with recursion that converts celsius to fahrenheit

```
#lang racket

(define (celsius-to-fahrenheit celsius)
  ;; Base case: If celsius is 0, return 32 (freezing point in Fahrenheit)
  (if (= celsius 0)
      32
      ;; Recursive case: Convert the remaining celsius and add 32
      (+ (* 9/5 celsius) 32)))

;; Example usage:
(displayln (celsius-to-fahrenheit 0)) ; Output: 32
(displayln (celsius-to-fahrenheit 100)) ; Output: 212
(displayln (celsius-to-fahrenheit -40)) ; Output: -40
```

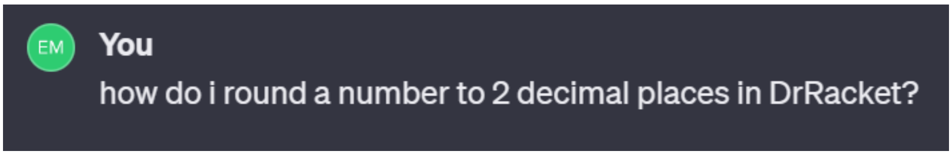
Prompt 4



write a DrRacket function with recursion that converts fahrenheit to celsius

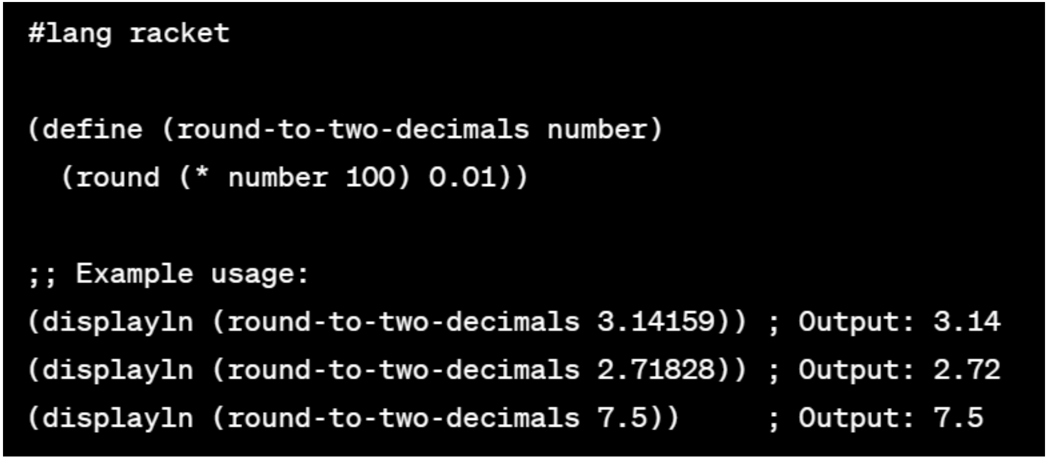
```
#lang racket

(define (fahrenheit-to-celsius fahrenheit)
  ;; Base case: If fahrenheit is 32, return 0 (freezing point in Celsius)
  (if (= fahrenheit 32)
      0
      ;; Recursive case: Convert the remaining fahrenheit
      (/ (* (- fahrenheit 32) 5/9) 1)))

;; Example usage:
(displayln (fahrenheit-to-celsius 32)) ; Output: 0
(displayln (fahrenheit-to-celsius 212)) ; Output: 100
(displayln (fahrenheit-to-celsius -40)) ; Output: -40
```

Prompt 5



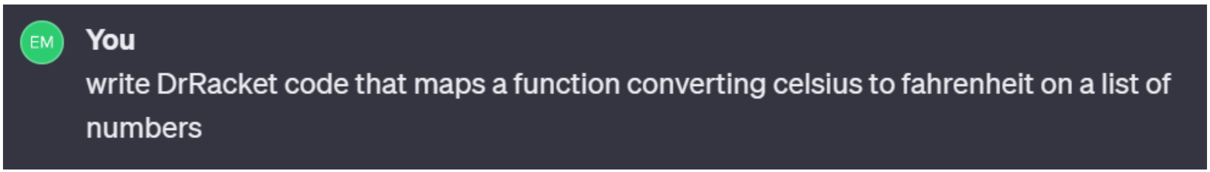
You

how do i round a number to 2 decimal places in DrRacket?



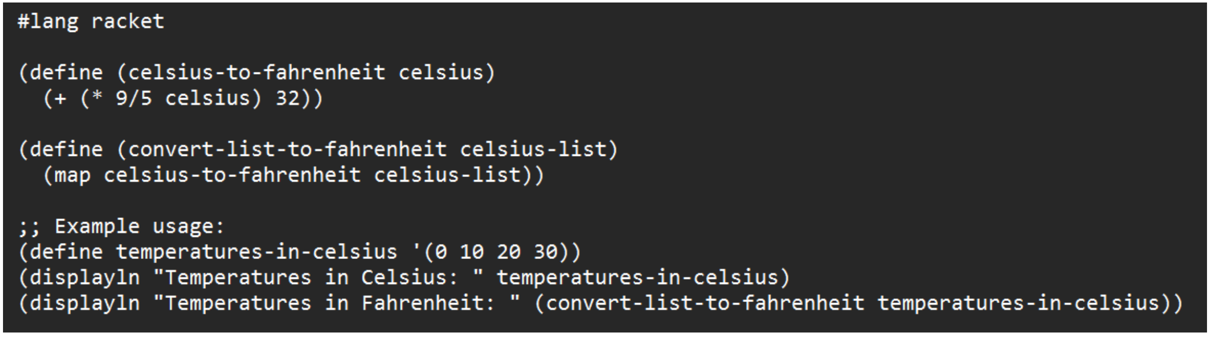
```
#lang racket

(define (round-to-two-decimals number)
  (round (* number 100) 0.01))

;; Example usage:
(displayln (round-to-two-decimals 3.14159)) ; Output: 3.14
(displayln (round-to-two-decimals 2.71828)) ; Output: 2.72
(displayln (round-to-two-decimals 7.5))     ; Output: 7.5
```

Prompt 6



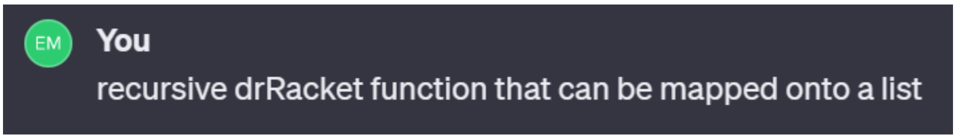
You

write DrRacket code that maps a function converting celsius to fahrenheit on a list of numbers



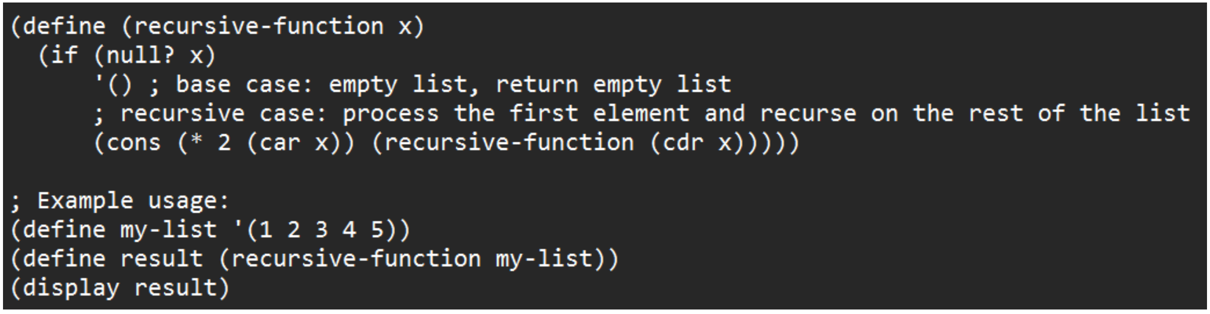
```
#lang racket

(define (celsius-to-fahrenheit celsius)
  (+ (* 9/5 celsius) 32))

(define (convert-list-to-fahrenheit celsius-list)
  (map celsius-to-fahrenheit celsius-list))

;; Example usage:
(define temperatures-in-celsius '(0 10 20 30))
(displayln "Temperatures in Celsius: " temperatures-in-celsius)
(displayln "Temperatures in Fahrenheit: " (convert-list-to-fahrenheit temperatures-in-celsius))
```

Prompt 7



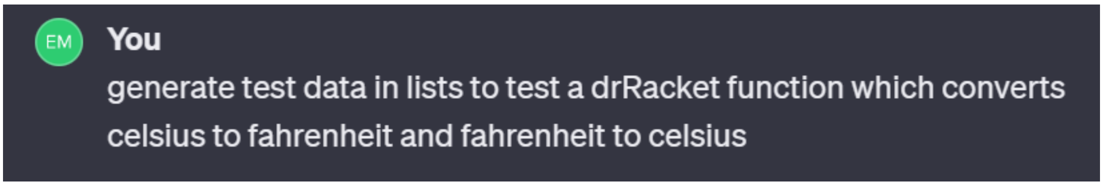
You

recursive drRacket function that can be mapped onto a list



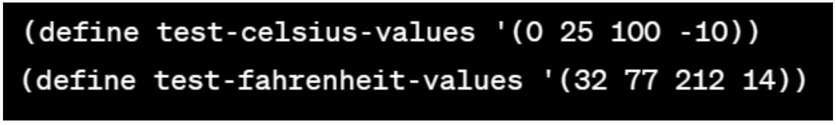
```
(define (recursive-function x)
  (if (null? x)
      '() ; base case: empty list, return empty list
      ; recursive case: process the first element and recurse on the rest of the list
      (cons (* 2 (car x)) (recursive-function (cdr x)))))

; Example usage:
(define my-list '(1 2 3 4 5))
(define result (recursive-function my-list))
(display result)
```

Prompt 8



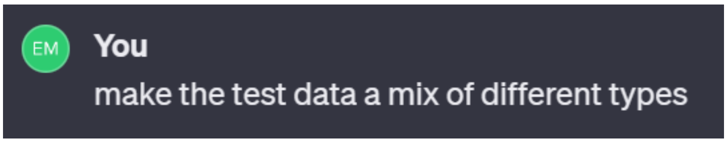
You

generate test data in lists to test a drRacket function which converts celsius to fahrenheit and fahrenheit to celsius



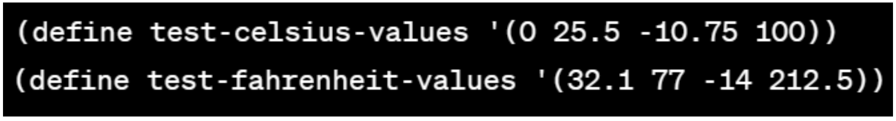
```
(define test-celsius-values '(0 25 100 -10))
(define test-fahrenheit-values '(32 77 212 14))
```

Prompt 9



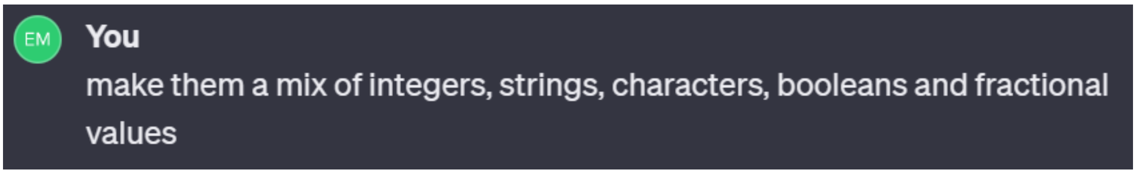
You

make the test data a mix of different types



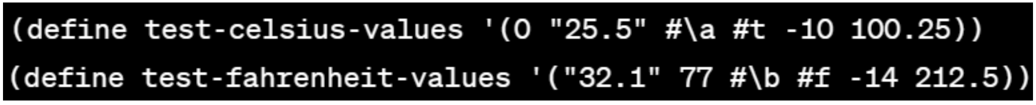
```
(define test-celsius-values '(0 25.5 -10.75 100))
(define test-fahrenheit-values '(32.1 77 -14 212.5))
```

Prompt 10



You

make them a mix of integers, strings, characters, booleans and fractional values



```
(define test-celsius-values '(0 "25.5" #\a #t -10 100.25))
(define test-fahrenheit-values '("32.1" 77 #\b #f -14 212.5))
```