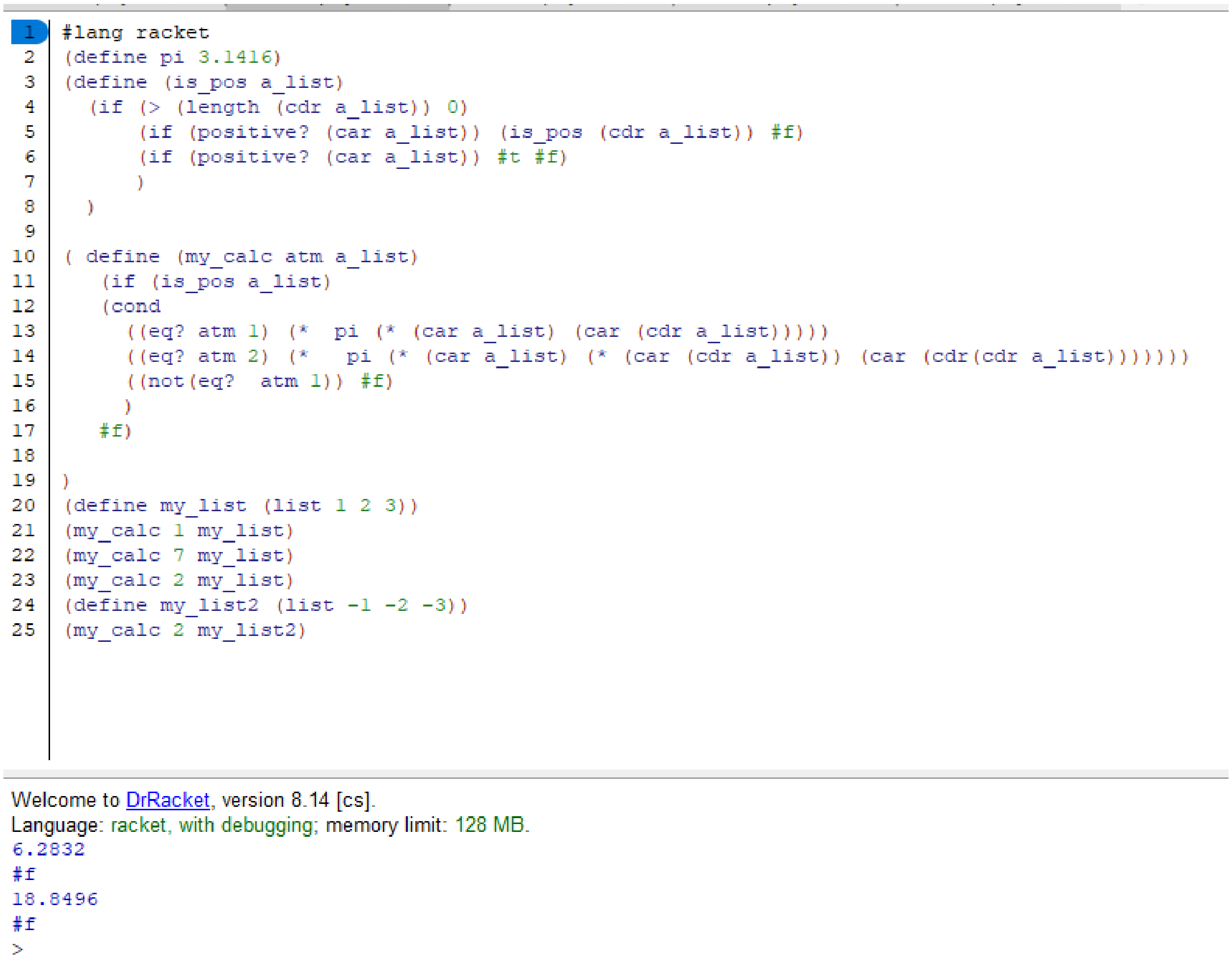
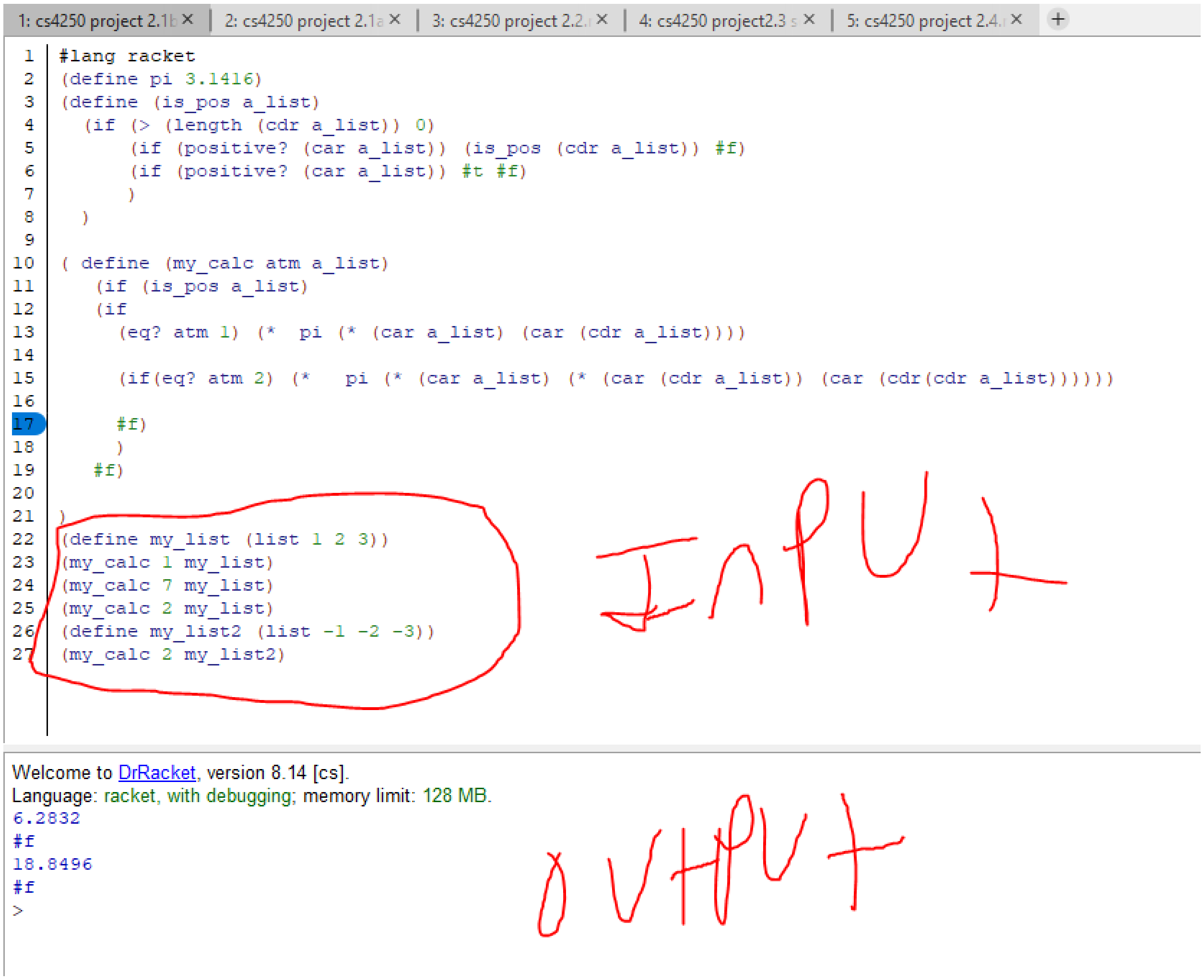
Problem 1 Part A



```
#lang racket
(define pi 3.1416)
(define (is_pos a_list)
  (if (> (length (cdr a_list)) 0)
      (if (positive? (car a_list)) (is_pos (cdr a_list)) #f)
      (if (positive? (car a_list)) #t #f)
      )
  )

( define (my_calc atm a_list)
   (if (is_pos a_list)
   (cond
     ((eq? atm 1) (*  pi (* (car a_list) (car (cdr a_list)))))
     ((eq? atm 2) (*   pi (* (car a_list) (* (car (cdr a_list)) (car (cdr(cdr a_list)))))))
     ((not(eq?  atm 1)) #f)
     )
   #f)

)
(define my_list (list 1 2 3))
(my_calc 1 my_list)
(my_calc 7 my_list)
(my_calc 2 my_list)
(define my_list2 (list -1 -2 -3))
(my_calc 2 my_list2)
```

```
Welcome to DrRacket, version 8.14 [cs].
Language: racket, with debugging; memory limit: 128 MB.
6.2832
#f
18.8496
#f
>
```

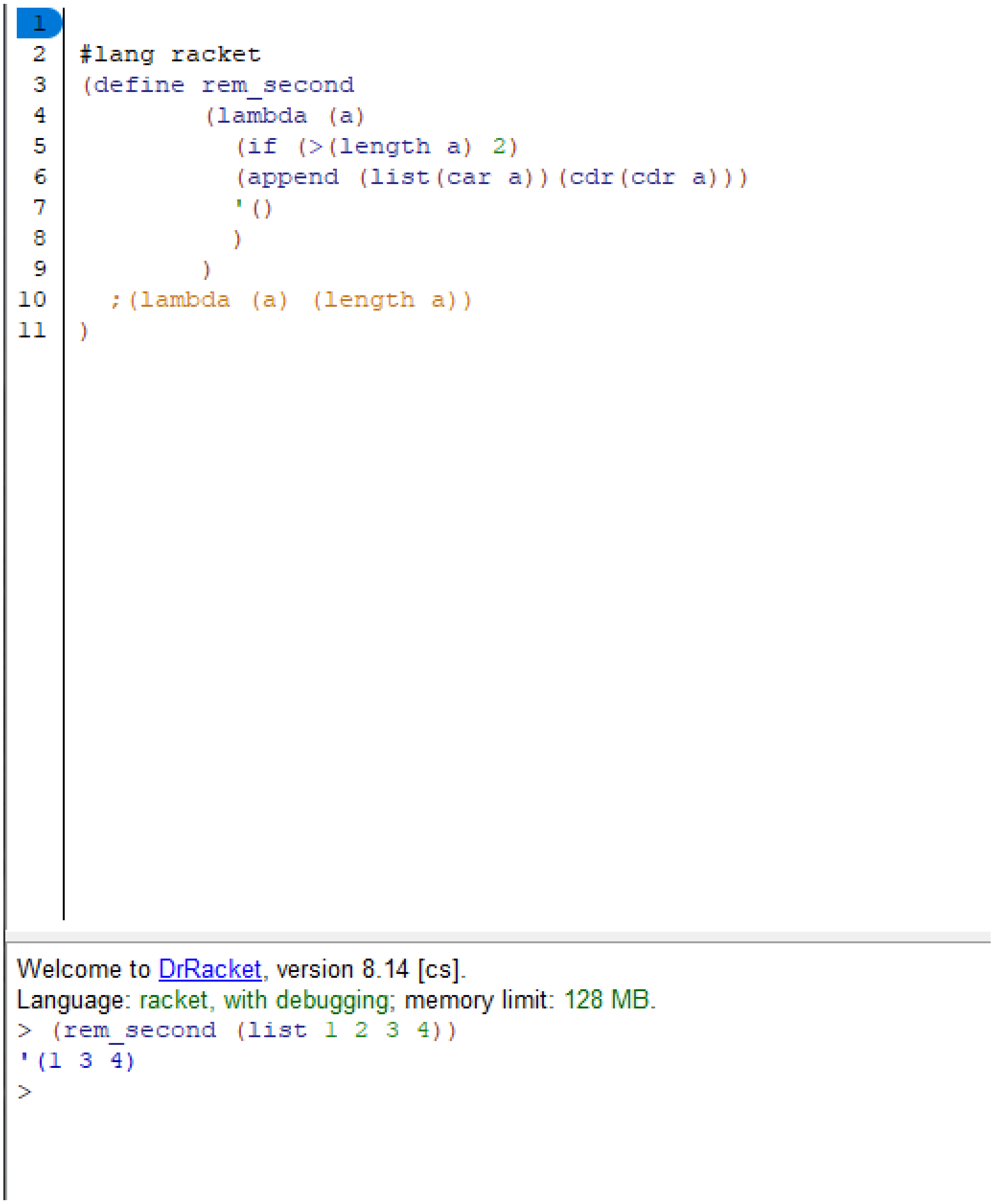
Problem 1 part B



```
#lang racket
(define pi 3.1416)
(define (is_pos a_list)
  (if (> (length (cdr a_list)) 0)
      (if (positive? (car a_list)) (is_pos (cdr a_list)) #f)
      (if (positive? (car a_list)) #t #f)
      )
  )

( define (my_calc atm a_list)
   (if (is_pos a_list)
   (if
     (eq? atm 1) (*  pi (* (car a_list) (car (cdr a_list))))

     (if(eq? atm 2) (*   pi (* (car a_list) (* (car (cdr a_list)) (car (cdr(cdr a_list))))))

     #f)
     )
   #f)

)
(define my_list (list 1 2 3))
(my_calc 1 my_list)
(my_calc 7 my_list)
(my_calc 2 my_list)
(define my_list2 (list -1 -2 -3))
(my_calc 2 my_list2)
```

```
Welcome to DrRacket, version 8.14 [cs].
Language: racket, with debugging; memory limit: 128 MB.
6.2832
#f
18.8496
#f
>
```

Problem 2

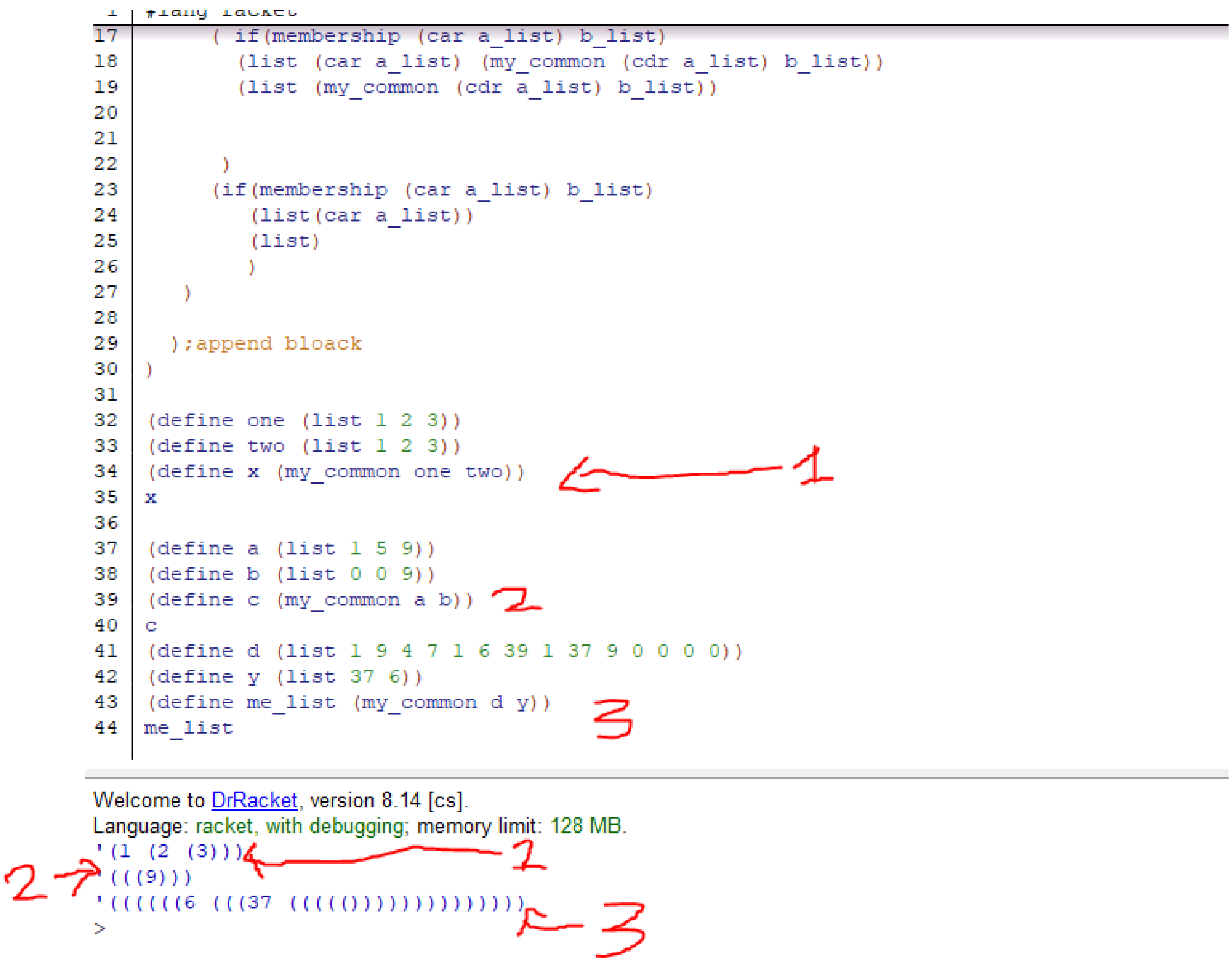


```
#lang racket
(define rem_second
        (lambda (a)
          (if (>(length a) 2)
          (append (list(car a))(cdr(cdr a)))
          '()
          )
        )
  ;(lambda (a) (length a))
)
```

```
Welcome to DrRacket, version 8.14 [cs].
Language: racket, with debugging; memory limit: 128 MB.
> (rem_second (list 1 2 3 4))
'(1 3 4)
>
```

Problem 3

(Ran into the problem that a lot of the nested parentheses remained, but the answer is still 100% accurate)

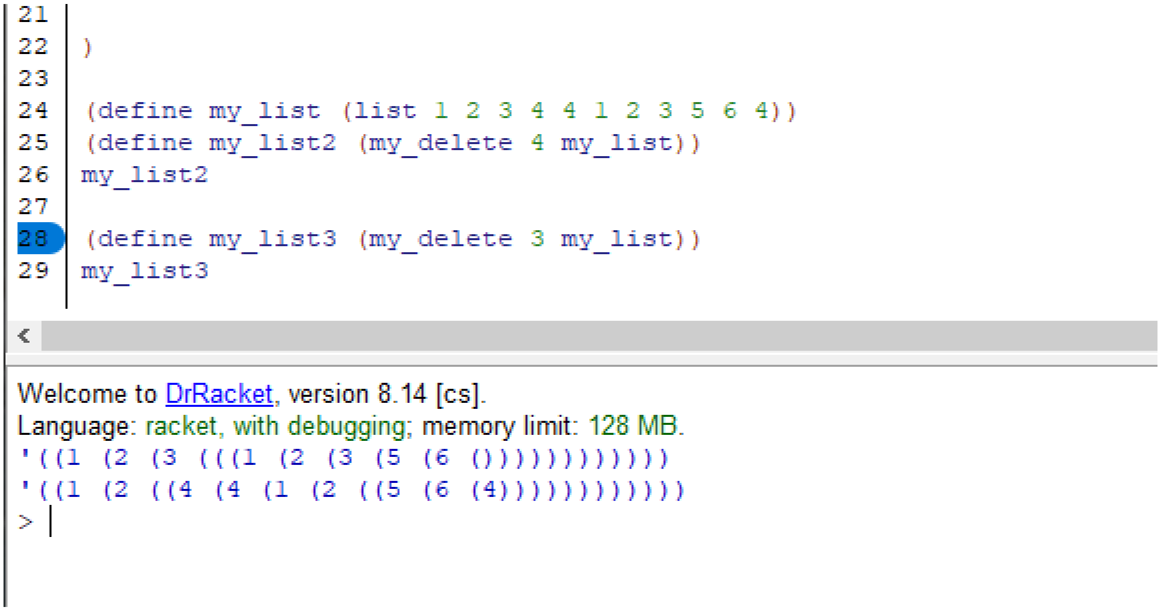


```
#lang racket
     ( if(membership (car a_list) b_list)
       (list (car a_list) (my_common (cdr a_list) b_list))
       (list (my_common (cdr a_list) b_list))


      )
     (if(membership (car a_list) b_list)
        (list(car a_list))
        (list)
        )
   )

  );append bloack
)

(define one (list 1 2 3))
(define two (list 1 2 3))
(define x (my_common one two))
x

(define a (list 1 5 9))
(define b (list 0 0 9))
(define c (my_common a b))
c
(define d (list 1 9 4 7 1 6 39 1 37 9 0 0 0 0))
(define y (list 37 6))
(define me_list (my_common d y))
me_list
```

```
Welcome to DrRacket, version 8.14 [cs].
Language: racket, with debugging; memory limit: 128 MB.
'(1 (2 (3)))
'(((9)))
'((((((6 (((37 ((((()))))))))))))))
>
```

Problem 4

(Same problem as last one but it works!)



```
)

(define my_list (list 1 2 3 4 4 1 2 3 5 6 4))
(define my_list2 (my_delete 4 my_list))
my_list2

(define my_list3 (my_delete 3 my_list))
my_list3
```

```
Welcome to DrRacket, version 8.14 [cs].
Language: racket, with debugging; memory limit: 128 MB.
'((1 (2 (3 (((1 (2 (3 (5 (6 ())))))))))))))
'((1 (2 ((4 (4 (1 (2 ((5 (6 (4)))))))))))))
> 
```