

## Problem 1 Part A

```
1 #lang racket
2 (define pi 3.1416)
3 (define (is_pos a_list)
4   (if (> (length (cdr a_list)) 0)
5       (if (positive? (car a_list)) (is_pos (cdr a_list)) #f)
6       (if (positive? (car a_list)) #t #f)
7   )
8 )
9
10 (define (my_calc atm a_list)
11   (if (is_pos a_list)
12       (cond
13         ((eq? atm 1) (* pi (* (car a_list) (car (cdr a_list)))))
14         ((eq? atm 2) (* pi (* (car a_list) (* (car (cdr a_list)) (car (cdr (cdr a_list)))))))
15         ((not(eq? atm 1)) #f)
16       )
17       #f)
18 )
19
20 (define my_list (list 1 2 3))
21 (my_calc 1 my_list)
22 (my_calc 7 my_list)
23 (my_calc 2 my_list)
24 (define my_list2 (list -1 -2 -3))
25 (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

```
1: cs4250 project 2.1.1 × | 2: cs4250 project 2.1.8 × | 3: cs4250 project 2.2.1 × | 4: cs4250 project 2.3.5 × | 5: cs4250 project 2.4.1 × | +
```

```
1 #lang racket
2 (define pi 3.1416)
3 (define (is_pos a_list)
4   (if (> (length (cdr a_list)) 0)
5       (if (positive? (car a_list)) (is_pos (cdr a_list)) #f)
6       (if (positive? (car a_list)) #t #f)
7   )
8 )
9
10 (define (my_calc atm a_list)
11   (if (is_pos a_list)
12       (if
13         (eq? atm 1) (* pi (* (car a_list) (car (cdr a_list))))
14         (if (eq? atm 2) (* pi (* (car a_list) (* (car (cdr a_list)) (car (cdr (cdr a_list))))))
15             #f)
16       )
17       #f)
18 )
19
20
21 )
22 (define my_list (list 1 2 3))
23 (my_calc 1 my_list)
24 (my_calc 7 my_list)
25 (my_calc 2 my_list)
26 (define my_list2 (list -1 -2 -3))
27 (my_calc 2 my_list2)
```

INPUT

---

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

OUTPUT

## Problem 2

```
1
2 #lang racket
3 (define rem_second
4   (lambda (a)
5     (if (>(length a) 2)
6       (append (list(car a)) (cdr(cdr a)))
7       '())
8     )
9   )
10 ; (lambda (a) (length a))
11 )
```

---

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 )

```

17 ( if(membership (car a_list) b_list)
18   (list (car a_list) (my_common (cdr a_list) b_list))
19   (list (my_common (cdr a_list) b_list))
20
21
22 )
23 (if(membership (car a_list) b_list)
24   (list(car a_list))
25   (list)
26 )
27 )
28
29 );append block
30 )
31
32 (define one (list 1 2 3))
33 (define two (list 1 2 3))
34 (define x (my_common one two)) ← 1
35 x
36
37 (define a (list 1 5 9))
38 (define b (list 0 0 9))
39 (define c (my_common a b)) 2
40 c
41 (define d (list 1 9 4 7 1 6 39 1 37 9 0 0 0 0))
42 (define y (list 37 6))
43 (define me_list (my_common d y)) 3
44 me_list

```

Welcome to [DrRacket](#), version 8.14 [cs].  
 Language: racket, with debugging; memory limit: 128 MB.

```

2 → '(1 (2 (3))) ← 1
      (((9)))
      '((((6 (((37 (((((((((((((((((((
>                                     3

```

#### Problem 4

(Same problem as last one but it works!)

```
21 |
22 | )
23 |
24 | (define my_list (list 1 2 3 4 4 1 2 3 5 6 4))
25 | (define my_list2 (my_delete 4 my_list))
26 | my_list2
27 |
28 | (define my_list3 (my_delete 3 my_list))
29 | 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))))))))))

> |