

# Practice Quiz 1a

**Due** No due date

**Points** 100


**Questions** 15

**Time limit** 50 Minutes

**Allowed attempts** Unlimited

## Instructions

Please note that during the real exam, there will be a paper copy of the Quiz Glossary available but you can also just pull it up as a separate tab in the Lockdown Browser.

[Quiz Glossary \(THIS LINK WORKS IN THE LOCKDOWN BROWSER\)](https://bain-cs111.github.io/course-files/quizzes/q3_glossary_compact.pdf)  [\(https://bain-cs111.github.io/course-files/quizzes/q3\\_glossary\\_compact.pdf\)](https://bain-cs111.github.io/course-files/quizzes/q3_glossary_compact.pdf)

<These are the exact instructions that will be on the real Quiz! They're just provided for reference here and you can ignore them for the practice quizzes.>

- This exam is ONLY to be taken in Tech Auditorium. Please do not open this exam from any other location.
- Please remember that you only get 1 attempt on this exam. You will have 50 minutes to complete it from the time that you start.

In part 1, if you get the answer *\*exactly\** right, canvas will mark it as CORRECT automatically. If there is any slight variation it will mark it as INCORRECT. After the exam is over, we will be going through each question and doing manual grading. For example, if you write (list of str) and canvas expected (listof string), canvas will automatically mark your answer wrong. However, when we go in to manually grade, we would mark the above answer as fully correct.

In part 2 and part 3, we will be grading the answers completely by hand. What we are looking for is that you understand what was wrong with the code and know how to fix it. We will utilize the line number(s) as a reference point, so that you can indicate where the error occurred, but more important will be your rewrite. Feel free to provide a written explanation for what is wrong in the code and how you will fix it. We will grade these with the same leniency and partial credit that we would in a printed exam.

If you encounter any issues while taking the exam, please raise your hand!

Take the quiz again

## Attempt history

	Attempt	Time	Score
LATEST	<a href="#">Attempt 1</a>	15 minutes	45 out of 100 *

\* Some questions not yet graded

Submitted 15 Oct at 23:39

Give the type of each of the following expressions. If more than one expression is given, assume that each one is executed, in order, and give the type of the value for the **last** expression.

- If it is a **primitive type** such as a number, string, Boolean or image (picture), just give the type.
- If it is a **record type (struct)**, just give the name of the record type. For example, if it's an album object, just say "album"
- If it is a **list**
  - If all the elements of the list are the same type, say "(listof *type*)" where *type* is the type of data in the list. For example (list 1 2 3) is a (listof number).
  - If it is a with different types of data, say (listof any)
  - If you know the result is specifically the empty list, which has no elements and therefore no element type, just say empty list.
  - If you know the result is a **list** but you don't know the type of data in it, just say "list" and we will give you partial credit.
- If the result is a **function**, give its argument and return types. That is, write the type(s) of its argument(s) followed by an arrow and the type of its result. If the procedure accepts any type of value for an argument, just say "any". For example:
  - The type of the **sin** function is: number -> number
  - The type of the **integer?** function is: any -> boolean
  - The type of the **<** function is: number number -> boolean
  - The type of the **square** function is: number string color -> image
- If you know the expression's value is a function but don't know its argument or return types, just say **function**, and we will give you partial credit.
- If executing it would produce an **exception**, say "Exception." You don't need to specify the type of exception.

### Question 1

5 / 5 pts

```
(ellipse (+ 1 1))
```

Correct!

Exception

Correct Answers

exception

### Question 2

5 / 5 pts

```
/ ; (i.e. a forward slash)
```

Correct!

number number -> number

Correct Answers

number number -> number

### Question 3

5 / 5 pts

```
(overlay square 10 "solid" "blue"  
          (square 20 "solid" "red"))
```

Correct!

Exception

Correct Answers

exception

#### Question 4

5 / 5 pts

(1 + 1)

Correct!

Exception

Correct Answers

exception

#### Question 5

5 / 5 pts

"bla bla bla"

Correct!

string

Correct Answers

string

#### Question 6

5 / 5 pts

```
(define proc  
  (λ (n) (λ (m) (+ m n))))  
(define f (proc 10))  
(f 20)
```

Correct!

number

Correct Answers

number

### Question 7

5 / 5 pts

```
(iterated-overlay (λ (n) (square 10 "solid" "blue"))  
                  10)
```

Correct!

image

Correct Answers

image

### Question 8

5 / 5 pts

```
(λ (n) 10)
```

Correct!

any -> number

Correct Answers

any -> number

### Question 9

0 / 5 pts

```
(iterated-overlay (iterated-overlay (λ (n) (rectangle n  
m  
"outline"  
"black"))
```

10) 10)

ou Answered

image

orrect Answers

exception

### Question 10

5 / 5 pts

`((λ (n) 10) +)`

Correct!

number

orrect Answers

number

Each of the following questions shows some code being executed (with a line number on the left of each line) at the Racket prompt, along with the output or error it generated, and the intended output that the programmer wanted. Give the **correction** to the code to produce the desired result.

**It is highly recommended that you copy and paste the code and then correct it as your answer (you can leave the line numbers in). If you correctly identify the error and fix it you will get full credit.**

What we are looking for is that you understand what was wrong with the code and know how to fix it. If you are unsure of what the problem is, then you can provide an explanation of your thoughts. We will accept the line number(s) as a reference point, so that you can indicate where the error occurred, but more important will be your rewrite. We will grade these with the same leniency and partial credit that we would in a printed exam.

## Question 11

Not yet graded / 10 pts

### Interaction

```
1 (+ (list 1 2 3 4 5 6))
```

### Actual Output:

```
+: expects a number, given #list<1 2 3 4 5 6>
```

### Desired Output:

the sum of the elements in the list (i.e. 21)

Your answer:

```
1 (apply + (list 1 2 3 4 5 6))
```

The problem is that `+` is being called with a list of numbers rather than with the numbers as separate arguments. There are two different ways of doing this:

```
(apply + (list 1 2 3 4 5 6))
```

```
(foldl + 0 (list 1 2 3 4 5 6))
```

Either will work.

## Question 12

Not yet graded / 10 pts

### Interaction

```
1 (map odd? (list 1 2 3 4 5 6 7))
```

### Actual Output:

```
#list<#true #false #true #false #true #false #true>
```

### Desired Output:

the odd numbers of the list, i.e. `#list<1 3 5 7>`

Your answer:

```
1 (filter odd? (list 1 2 3 4 5 6 7))
```

```
(filter odd? (list 1 2 3 4 5 6 7))
```

The programmer said map when they meant filter.

## Question 13

Not yet graded / 10 pts

### Interaction

```
1 (λ (n)
2   (iterated-beside (λ (n)
3                     (square 20 "outline" "black"))
4                     10))
```

### Actual Output:

```
(lambda (a1) ...)
```

### Desired Output:



--	--	--	--	--	--	--	--	--	--

Your answer:

```
1 (iterated-beside (λ (n)
2                   (square 20 "outline" "black"))
3                   10)
```

```
(iterated-beside (λ (n)
                  (square 20 "outline" "black"))
                  10)
```

The programmer had wrapped the code in a gratuitous  $\lambda$  expression.

## Question 14

Not yet graded / 10 pts

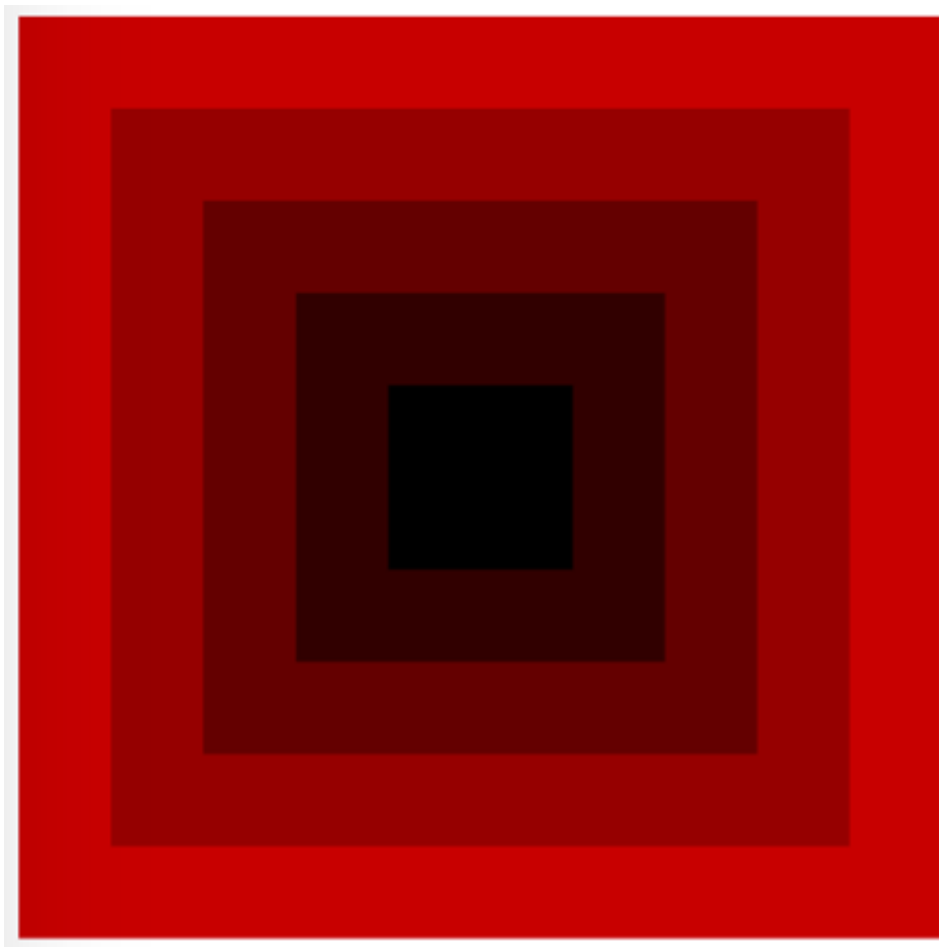
### Interaction

```
1 (iterated-overlay (λ (n)
2                   (square (* + n 1)
3                           50)
4                   "solid"
5                   (color (* n 50)
6                           0
7                           0))
8                   5)
```

### Actual Output:

lambda: expected only one expression for the function body, but found 2 extra parts

### Desired Output:



Your answer:

[illegible]

```
(iterated-overlay (λ (n)
  (square (* (+ n 1) 50)
    "solid"
    (color (* n 50)
      0
      0)))
5)
```

The following questions show a function definition. In the blank below, please provide one valid test (`check-expect`) of the procedure.

### Question 15

Not yet graded / 10 pts

```
; sum: (listof number) -> number  
; returns the sum of the items in the list  
(define sum (lambda (the-list)  
  (foldl + 0 the-list)))
```

In the blank below, please provide one valid test (`check-expect`) of the procedure.

Your answer:

(`check-expect (sum (list 1 2 3 4)) 10`)

One possible answer.

(`check-expect (sum (list 1 2 3 4)) 10`)