[REAL, GRADED] Quiz 2- Requires Respondus LockDown Browser

Due 1 Nov at 13:55 **Points** 100 **Questions** 16
Available until 1 Nov at 13:55 **Time limit** 50 Minutes
Requires Respondus LockDown Browser

Instructions

This exam is ONLY to be taken in Tech Auditorium during your scheduled exam time. Any other testing environment without explicit permission from the instructor will be considered a violation of our academic honesty policy.

- 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 unless you have testing accommodations that allow for extra time.
- **Quiz Glossary (THIS LINK WORKS IN THE LOCKDOWN BROWSER) (https://bain-cs111.github.io/course-files/quizzes/q3_glossary_compact.pdf)**

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 can utilize 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.

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

This quiz was locked 1 Nov at 13:55.

Attempt history

	Attempt	Time	Score
LATEST	Attempt 1	50 minutes	97 out of 100

Score for this quiz: **97** out of 100
Submitted 1 Nov at 13:51
This attempt took 50 minutes.

Question 1

0 / 0 pts

RULES

1) You may not communicate with anyone (electronically or in person) during this exam. You must turn off your phone and other nearby devices after successfully logging into Canvas.

2) You may not use any resources during this exam. You may only use your computer in the Lockdown browser. You may not use any printed/written notes/books or any other devices/tools while taking this exam with TWO EXCEPTIONS:

- You may use a printed Quiz glossary (remember you can use the electronic version by clicking on the link provided in each set of instructions and opening it in a new tab).
- You may use scratch paper and a pen/pencil to take notes and sketch out solutions.

****

The following statement is an excerpt from the Northwestern Provost's Office "Academic Integrity: A Basic Guide"

A. Basic Standards of Academic Integrity

Registration at Northwestern requires adherence to the University's standards of academic integrity. These standards may be intuitively understood, and cannot in any case be listed exhaustively; the following examples represent some basic types of behavior that are unacceptable:

1. **Cheating:** using unauthorized notes, study aids, or information on an examination; altering a graded work after it has been returned, then submitting the work for regrading; allowing another person to do one's work and submitting that work under one's own name; submitting identical or similar papers for credit in more than one course without prior permission from the course instructors.

2. **Plagiarism:** submitting material that in part or whole is not entirely one's own work without attributing those same portions to their correct source.
3. **Fabrication:** falsifying or inventing any information, data or citation; presenting data that were not gathered in accordance with standard guidelines defining the appropriate methods for collecting or generating data and failing to include an accurate account of the method by which the data were gathered or collected.
4. **Obtaining an Unfair Advantage:** (a) stealing, reproducing, circulating or otherwise gaining access to examination materials prior to the time authorized by the instructor; (b) stealing, destroying, defacing or concealing library materials with the purpose of depriving others of their use; (c) unauthorized collaborating on an academic assignment (d) retaining, possessing, using or circulating previously given examination materials, where those materials clearly indicate that they are to be returned to the instructor at the conclusion of the examination; (e) intentionally obstructing or interfering with another student's academic work (f) recycling one’s own work done in previous classes without obtaining permission from one’s current instructor or (g) otherwise undertaking activity with the purpose of creating or obtaining an unfair academic advantage over other students' academic work.
5. **Aiding and Abetting Academic Dishonesty:** (a) providing material, information, or other assistance to another person with knowledge that such aid could be used in any of the violations stated above; (b)providing false information in connection with any inquiry regarding academic integrity; or (c) providing(including selling) class materials to websites that sell or otherwise share such materials – including homework, exams and exam solutions, submitted papers or projects, as well as original course materials (for example, note packets, power point decks, etc.). In addition to violating Northwestern’s policies on academic integrity, such conduct may also violate University policies related to copyright protection.
6. **Falsification of Records and Official Documents:** altering documents affecting academic records; forging signatures of authorization or falsifying information on an official academic document, grade report, letter of permission, petition, drop/add form, ID card, or any other official University document.
7. **Unauthorized Access to computerized academic or administrative records or systems:** viewing or altering computer

records, modifying computer programs or systems, releasing or dispensing information gained via unauthorized access, or interfering with the use or availability of computer systems or information.

If you agree to follow these rules, please type "I agree" in the following blank

Correct!

I agree

orrect Answers

I agree

"I agree"

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 function 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. This includes a program that will run out of memory due to infinite recursion.

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

Question 2

5 / 5 pts

```
(append (list 1 2) (list 3 4) 5)
```

Correct!

exception

orrect Answers

exception

Question 3

5 / 5 pts

```
(define (a-function a-list)
  (local [(define (loop a-list accum)
            (if (empty? a-list)
                accum
                (loop (rest a-list)
                      (+ accum 1))))]
    (loop a-list 0)))
(a-function (list 1 "2" 3))
```

Correct!

number

orrect Answers

number

3

num

Question 4

5 / 5 pts

```
(define (an-action a-thing)
  (if (empty? a-thing)
      '()
      (if (= 1 (first a-thing))
          (cons (first a-thing)
                (an-action (rest a-thing)))
          (an-action (rest a-thing)))))
an-action
```

Correct!

(listof number) -> (listof number)

orrect Answers

(listof number) -> (listof number)

Question 5

5 / 5 pts

```
(define (pineapple an-input)
  (if (empty? an-input)
      0
      (+ 1
         (pineapple (rest an-input)))))

pineapple
```

Correct!

(listof any) -> number

orrect Answers

(listof any) -> num

(listof any) -> number

Question 6

5 / 5 pts

```
(foldl cons '() '(1 2 3 4 5))
```

Correct!

(listof number)

orrect Answers

(listof number)

(listof nums)

(listof num)

listof numbers

Question 7

5 / 5 pts

```
(cons "happy" (cons "birthday" (cons "to..." '())))
```

Correct!

(listof string)

orrect Answers

(listof string)

(listof strings)

Question 8

5 / 5 pts

```
(define a 50)
(local [(define a "blue")]
  (rectangle (* 5 10) 50 "solid" a))
```

Correct!

image

orrect Answers

picture

image

Question 9

5 / 5 pts

```
(define (im-just-ken thing-1 thing-2)
  (local [(define (barbie f x accum)
            (if (f (first x))
                (barbie f (rest x) (cons (first x) accum))
                (barbie f (rest x) accum)))]
    (barbie thing-1 thing-2 '())))

(im-just-ken odd? '(1 2 3 4 5))
; Hint: make sure to consider the end of the recursion.
```

Correct!

exception

orrect Answers

exception

Question 10

5 / 5 pts

```
(define x (string-length "kenobi"))
(cond  [(= x 1)   "hello"]
       [(= x 2)   "there"]
       [(= x 3)   (list "g" "e" "n" "e" "r" "a" "l")]
       [else      #false])
```

Correct!

boolean

orrect Answers

false

boolean

bool

#f

#false

Question 11

5 / 5 pts

```
; A human is either...
; - (make-human string human human)
; - just a name (string)
(define-struct human (name parent-1 parent-2))
(define test-humans (list (make-human "Tony Stark"  "Maria"  "Howard")
                          (make-human "Bruce Wayne" "Martha" "Thomas")))
(map human-parent-1 test-humans)
```

Correct!

(listof string)

orrect Answers

(listof strings)

(listof human)

(listof humans)

(listof string)

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. Remember, **you are not allowed to completely rewrite the given program.**

It is highly recommended that you copy and paste the code and then correct it as your answer. If you correctly identify the error and fix it you will get full credit. When copy and pasting, you can highlight just the code without the line numbers.

If you'd like your code to look more like Racket code, you can use the Canvas Toolbar to select the PREFORMATTED style. In the toolbar for the answer box, you'll see a button that says "Paragraph" with a down arrow (right next to the font size). Click on that then select Preformatted which will format your Text to look more like Racket code.

If you are on a Chromebook that doesn't allow Ctrl-C then Ctrl-V, you can highlight the code and then "drag it" to the box below in order to copy.

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.

[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)

Question 12

10 / 10 pts

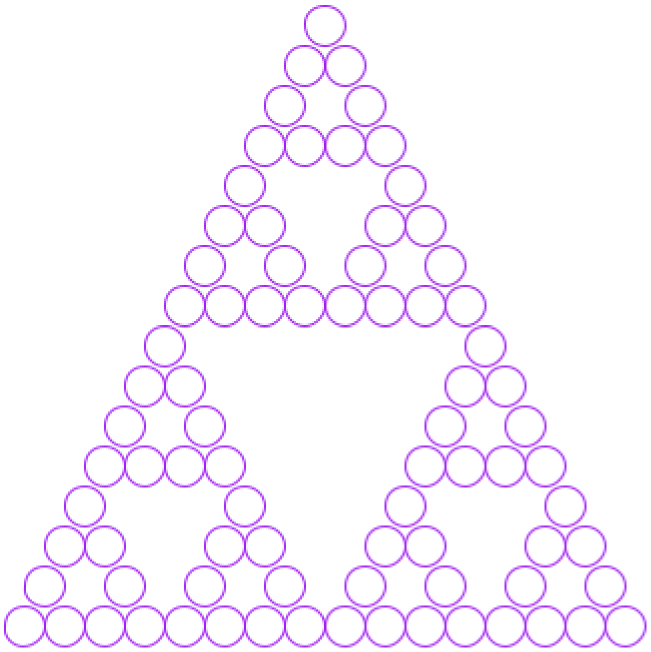
Interaction

```
; bubble-triangle: number -> image
; a function that takes in a number
; and builds an image with n levels.
(define (bubble-triangle n)
  (if (= n 0)
      (circle 10 "outline" "purple")
      (local [(define loop (bubble-triangle n))]
        (above loop
               (beside loop
                       loop)))))
(bubble-triangle 4)
```

Actual Output:

```
Your computer ran out of memory (runs forever).
```

Desired Output:


(a triangular fractal image made of purple circles should be rendered above. Please raise your hand if you don't see an image.

Your answer:

```
; bubble-triangle: number -> image
; a function that takes in a number
; and builds an image with n levels.
(define (bubble-triangle n)
  (if (= n 0)
      (circle 10 "outline" "purple")
      (local [(define loop (bubble-triangle (- n 1)))]
        (above loop
               (beside loop
                       loop)))))
(bubble-triangle 4)
```

Pay special attention to the recursive call here. Are we making the "solution" one step simpler each time?

:)

Question 13

10 / 10 pts

Interaction

```
; A tree is...
;  - (make-tree any (listof tree))
(define-struct tree (data child-list))

(define test-tree (make-tree 1
                             (list (make-tree 2 '())
                                   (make-tree 3 '())))))

; count-nodes: tree ->  number
; returns the number of nodes in a tree defined as above
(define (count-nodes a-tree)
  (+ 1
     (foldl + 0
            (map count-nodes
                 a-tree))))

(count-nodes test-tree)
```

Hint: Note that in this representation, there's no either/or in the definition, there's no representation for an empty tree, and a leaf is just a node with an empty child-list

Actual Output:

```
map: 2nd argument must be a list, given (make-tree 1 (list (make-tree 2 '())
(make-tree 3 '())))
```

Desired Output:

```
3 ; the number of nodes in this tree
```

Your answer:

```
; A tree is...
;  - (make-tree any (listof tree))
(define-struct tree (data child-list))

(define test-tree (make-tree 1
                             (list (make-
tree 2 '())
                                   (make-
tree 3 '())))))
```

```
; count-nodes: tree ->  number
; returns the number of nodes in a tree d
efined as above
(define (count-nodes a-tree)
  (+ 1
     (foldl + 0
            (map count-nodes
                 (tree-child-list a-tre
e)))))

(count-nodes test-tree)
```

Question 14

7 / 10 pts

Interaction

```
; just-strings: (listof any) -> (listof string)
; returns a list of all strings in a given list
(define (just-strings lst)
  (local [(define (help alst acc)
            (cond [(empty? alst)          acc]
                  [(string? (first alst)) (help (rest alst)
                                                acc)]
                  [else                   (help (rest alst)
                                                acc)]))]
    (reverse (help lst empty))))

(just-strings (list 1 "2" 3 "4"))
```

Note, `reverse` is a built-in function that takes in any list and returns the same list reversed. In other words `(reverse '(1 2 3))` outputs `'(3 2 1)`. Its type signature is `(listof any) -> (listof any)`.

Actual Output:

```
'()
```

Desired Output:

```
(list "2" "4") ; list containing just the strings from the original list.
```

Your answer:

```
; just-strings: (listof any) -> (listof string)
; returns a list of all strings in a given list
(define (just-strings lst)
  (local [(define (help alst acc)
            (cond [(empty? alst)          acc]
                  [(string? (first alst)) (cons (first alst) (help (re
st alst)
                                                                 acc))]
                  [else                   (help (rest alst)
                                                acc)]))]
    (reverse (help lst empty))))

(just-strings (list 1 "2" 3 "4"))
```

Pay careful attention to the accumulator in the helper function call...is it being updated?

Question 15

10 / 10 pts

Interaction

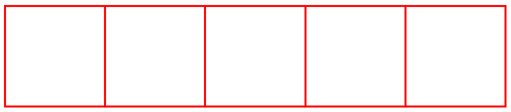
```
(define (iterated-any combiner gen num)
  (cond [(= num 0) empty-image]
        [else      (combiner (iterated-any combiner gen (- num 1))
                             (gen (- num 1)))]))

(iterated-any beside (square 50 "outline" "red") 5)
```

Recall that `iterated-any`, when given a function like `beside` as its first input, should behave identically to `iterated-beside`.

Actual Output:

```
gen (line 4): expected a function after the open parenthesis, but received #
<image>
```

Desired Output:



5 red outlined squares of size 50 placed beside each other should be rendered above

Your answer:

```
(define (iterated-any combiner gen num)
  (cond [(= num 0) empty-image]
        [else      (combiner (iterated-any combiner gen (- num 1))
                             (gen (- num 1)))]))

(iterated-any beside (lambda (n) (square 50 "outline" "red")) 5)
```

Look at how the 2nd input to iterated-any is used inside of the recursive call. Can we use it as an image? or does it need to be some other type of data?

The following questions show a function definition. In the blank below you **may be asked to provide multiple tests** (`check-expect`) of the function. That is you, need to come up with and write valid tests that assesses whether or not the function works as the programmer intends **and that cover the specific cases referenced in the question**.

If you'd like your code to look more like Racket code, you can use the Canvas Toolbar to select the PREFORMATTED style. In the toolbar for the answer box, you'll see a button that says "Paragraph" with a down arrow (right next to the font size). Click on that then select Preformatted which will format your Text to look more like Racket code.

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

Question 16

10 / 10 pts

```
; count-down: number -> (listof number)
; produce a list of numbers counting down from the input to 1
(define (count-down n)
  (cond [(= n 0) '()]
        [else (cons n
                    (count-down (- n 1)))]))

; friends-count-down: number -> (listof number)
; (the same as count-down)
; the same function as count-down, but provided by your friend
(define (friends-count-down n)
   ;——code redacted——
)
```

In the blank below, please provide **three** valid tests of the function that:

- **Tests the** `count-down`**'s base case.**
- **Tests a typical input to** `count-down`
- **Compares the result of** `count-down` **to a friend's version of the same function, defined above, that has the exact same input requirements as this function. You don't know how your friend has written their function but you do know that both functions should produce the same output for a given input.**

Your answer:

```
(check-expect (count-down 0)
              '())

(check-expect (count-down 3)
              (list 3 2 1))

(check-expect (count-down 4)
              (friends-count-down 4))
```

Quiz score: **97** out of 100