

Booleans and Conditionals

Exercises

Exercise 0: Preparation

You may find it helpful to rescan the readings on Boolean values and conditionals.

You may also want to rescan the reading on numbers.

After making sure that you're prepared, start DrScheme.

Exercise 1: Type Predicates

Fill in the following table (or as much as you think is appropriate)

	5	5.0	'five	"five"	list	#t	#f	(cons 'a null)	null	'null	()
number?											
symbol?											
string?											
procedure?											
boolean?											
list?											

Exercise 2: Empty lists

Which of the following does Scheme consider an empty list?

- null
- 'null
- ()
- (list a)
- (list)
- 'nothing

Exercise 3: Equality

Consider the following definitions

```
(define alpha (list 'a 'b 'c))  
(define beta (list 'a 'b 'c))  
(define gamma alpha)
```

Determine which of the lists are `eq?`, `eqv?`, or `equal?`.

Exercise 4: Reflection

Did you see anything surprising in the previous exercises?

Exercise 5: What is not?

- What type is `not`?
- What predicate would you use to verify your answer?

Exercise 6: What is not? Revisited

The symbol `not` is the *name* of something, as you determined in the preceding exercise. However, the symbol itself, considered as a datum, is not a procedure. Does Scheme agree with *this* classification? How could one ask Scheme whether the *symbol* `not` is a procedure?

Exercise 7: Combining Boolean Values

Fill in the following tables for each of the operations `and` and `or`.

`and`

First argument	Second argument	Result
#f	#f	
#f	#t	
#t	#f	
#t	#t	

`or`

First argument	Second argument	Result
#f	#f	
#f	#t	
#t	#f	
#t	#t	

Exercise 8: Ranges

- Write a Boolean expression that determines if the value named by `grade` is between 0 and 100, inclusive.
- Test that expression using different values of `grade`.

Exercise 9: Exploring `and` and `or`

- Determine the value `and` returns when called with no parameters.
- Explain why `and` returns that value.
- Determine the value `or` returns when called with no parameters.
- Explain why `or` returns that value.

Exercise 10: What is it?

Define and test a Scheme predicate `atom-or-list?` that takes one argument and returns `#t` if the argument is either an atom (symbol) or a list, `#f` if it is neither.

Exercise 11: Between, Revisited

Define and test a Scheme predicate `between?` that takes three arguments, all real numbers, and determines whether the second one lies strictly between the first and third (returning `#t` if it is, `#f` if it is not). For example, 6 lies strictly between 5 and 13, so both `(between? 5 6 13)` and `(between? 13 6 5)` should have the value `#t`.

Exercise 12: Triangulation

Three line segments can be assembled into a triangle if, and only if, the length of each of them is less than the sum of the lengths of the other two. Define a Scheme predicate `triangle?` that takes three arguments, all positive real numbers, and determines whether line segments of those three lengths (assumed to be measured in the same units) could be assembled into a triangle.

Exercise 13: Being Neighborly

Define and test a Scheme procedure `neighbor` that takes one argument, an integer, and returns the next higher integer if its argument is even, the next lower integer if its argument is odd. (Start by writing a comment that describes the purpose of the procedure.)

Exercise 14: Non-Boolean Tests

For each of the following expressions, guess what the output should be and then test it in Scheme.

- a. `(if #t 'aardvark 'zebra)`
- b. `(if #f 'aardvark 'zebra)`
- c. `(if (null? null) 'aardvark 'zebra)`
- d. `(if (null? 'null) 'aardvark 'zebra)`
- e. `(if (null? (list 'a 'b 'c)) 'aardvark 'zebra)`
- f. `(if () 'aardvark 'zebra)`
- g. `(if (list 'a 'b 'c) 'aardvark 'zebra)`
- h. `(if 2 'aardvark 'zebra)`
- i. `(if 'true 'aardvark 'zebra)`
- j. `(if 'false 'aardvark 'zebra)`

Exercise 15: Categorizing Lists

Define and test a Scheme procedure `list-type` that takes one argument, a list, and returns the symbol `empty` if the argument is the empty list, the symbol `non-empty` otherwise.

Exercise 16: Who Won?

Define and test a Scheme procedure `report-victory` that takes one argument, a real number, and returns the symbol `won` if that number is positive, the symbol `lost` if it is negative, and the string `tied` if it is zero.

- a. Use `if` for all the tests within your procedure.
- b. Use `cond` for all the tests within your procedure.

Exercise 17: The Sphinx's Riddle

As you may know, one of the famous riddles of the Sphinx goes something like the following:

What is it that walks upon four legs, then two legs, then three legs?

The answer is “humans”.

Write a Scheme predicate, `legs`, that, given someone's age, tells how many legs they walk upon. (You get to choose reasonable ages for the three phases of life.)

Exercise 18: Categorizing Data

Write and test a Scheme procedure, `type`, that returns the a symbol that represents the type of its argument. For example,

```
> (type 3)
integer
> (type '(1 2 3))
list
> (type "hello")
string
> (type 2/3)
rational
> (type 'type)
symbol
> (type type)
procedure
```

Notes

Notes on Problem 9

`(and)` has value `true` because “`and` has a value of `true` if none of the parameters have value `false`”. Since it has no parameters, none are `false`.

`(or)` has value `false` because “`or` has value `false` if none of the parameters is non-`false`”. Since it has no parameters, none are non-`false`.