This tutorial will cover the following main ideas:

- Tracing expressions involving nested lists using substitution rules.
- Solving problems involving two-dimensional data and association lists.
**Question 1: Tracing Nested Lists**

Here is a summary of the substitution rules that apply to this question:

1. Functions must be applied on values (i.e., all arguments must be values, not expressions)
2. Given a choice, evaluate expressions from left-to-right (or top-to-bottom)
3. \((\text{first } (\text{cons } a \ b)) \Rightarrow a\), where \(a\) and \(b\) are values.
4. \((\text{rest } (\text{cons } a \ b)) \Rightarrow b\), where \(a\) and \(b\) are values.

Finally, \((\text{second my-list})\) is shorthand for \((\text{first } (\text{rest my-list}))\) and produces the second element of \(\text{my-list}\).
Question 1: Tracing Nested Lists

(define intro-courses (list 'cs100
  (list (list (list 'cs114)'cs115)
    'cs116
    (list 'cs135))
  'cs105
  'cs106))

(first (second (rest (first (rest intro-courses))))))
Question 2: Verifying Battleship Grids

In the game of Battleship, players take turns shooting missiles into the ocean, trying to hit their opponent’s hidden battleships.

We represent the ocean as a square two-dimensional grid, where each space can either contain a battleship, a sunken battleship, or be empty. In this problem, all battleships are the same size and only take up one space in the grid. Each element in the grid is a symbol, one of:

- ' _ ' is an open square,
- 'B' is a small battleship,
- 'S' is a sunken battleship

1. Write a data definition for a battleship grid (BattleGrid), defined using nested lists.
2. Write a function (check? grid row column status) which determines whether the
given grid has the given status at the position (row, column). Both the row and
column coordinates must be within the bounds of the grid.

(check-expect (check? (list (list 'B '_)
               (list '_ 'S)) 0 0 'B) true)
We can also represent some or all positions in the grid using an association list, where each position (row, column) is mapped to its status (either ‘B, ‘S, or ‘_).

For example, this association list:
(define grid-bal (list (list (list 0 0) 'B)
                    (list (list 1 0) 'B)
                    (list (list 2 1) 'B)))

tracks all the positions of battleships (that haven’t yet been sunk) in this 3x3 grid:
(define grid (list (list 'B '_ '_)
                (list 'B '_ 'S)
                (list '_ 'B '_)))

1. Write a data definition for a battleship association list, Battle-AL, as described above.
Recall that an association list is just a list of (key, value) pairs, where each pair is stored as a two-element list. Module 8, slide 29 provides a data definition for an association list with natural numbers as keys and strings as values:

;;; An association list (AL) is one of:
;;; * empty
;;; * (cons (list Nat Str) AL)
;;; Requires: each key (Nat) is unique
2. Write the functions (row kv), (col kv), and (status kv) to extract the row, column, and status from a key-value pair in a BattleAL.

Reminder: In Module 8, slide 33, we have the following functions for extracting the key (natural number) and value (string) for the example association list.

(define (key kv) (first kv))
(define (val kv) (second kv))
3. Write the function (verify? grid b-al) which, for each position key in the battleship association list b-al, verifies that the corresponding position in the grid has the status value associated with that key.

```
(define grid-bal (list (list (list 0 0) 'B)
                        (list (list 1 0) 'B)
                        (list (list 2 1) 'B)))

(define grid (list (list 'B '_ '_)
                    (list 'B '_ 'S)
                    (list '_ 'B '_)))

(check-expect (verify? grid grid-bal) true)
```