CS 231
Naomi Nishimura

Module `equiv.py`

We can view two lists as **equivalent** if they contain exactly the same data items, even if the items appear in different orders. Extending this notion to lists of lists, the two lists must contain equivalent data items, which again can appear in different orders.

For example, the following lists are equivalent:

- [1, 2, 3, 3, 4, 4, 4]
- [4, 1, 2, 4, 3, 3, 4]

However, the list [1, 2, 3, 4] is not equivalent to either list, as it contains a different total number of data items.

Similarly, the following lists of lists are equivalent:

- [[1, 2], [3, 3, 4], [4, 5], [5, 4]]
- [[4, 5], [4, 3, 3], [4, 5], [2, 1]]

Each lists of lists contains four lists, one equivalent to [1, 2], one equivalent to [3, 3, 4] and two equivalent to [4, 5].

At times during the course, you will be writing functions that are correct when they produce any one of several equivalent lists (or lists of lists). In order to make it possible for you to write tests for such functions, I have provided a module `equiv.py` that contains the following three functions:

- `equiv(one, two)` produces `True` if the lists `one` and `two` are equivalent and `False` otherwise
- `equiv_lol(one, two)` produces `True` if the lists `one` and `two` contain the same lists (in any order), where lists are the same if they contain the same items in any order, and `False` otherwise
- `in_equiv(one, two)` produces `True` if the list of lists `two` contains a list that is equivalent to the list `one` and `False` otherwise

The file `equivuse.py` contains tests of the three functions, showing examples of their use.

You may wish to use the module in testing; use `from equiv import *` at the beginning of the file being tested. You can then use tests of the following forms, where `computed` is the result computed by the function you are testing and `expected` is the result that you are expecting to obtain:

```
check.expect("Test one", equiv(computed_one, expected_one), True)
check.expect("Test two", equiv_lol(computed_two, expected_two), True)
check.expect("Test three", in_equiv(computed_three, expected_three), True)
```