ICA-5: Invarant (35 min)

Problem 1. Consider the following function:

```
def concat_keys(dict_):
    string = ""
    for key in dict_.keys():
        # POINT 1
        string += str(key)

    # POINT 2
    return string
```

a) For each iteration, what can we say about the contents of `string` in this program at **POINT 1**? Describe your invariant in English.

b) What can we say about the contents of `string` in this program at **POINT 2**? Describe your invariant in English.

Problem 2. Consider the following function.

```
def func(n):
    outer_list = []
    for i in range(n):
        # POINT 1
        inner_list = []
        for j in range(n):
            inner_list.append(n)
        outer_list.append(inner_list)
    # POINT 2
    return outer_list
```

What invariant relationship holds between the value of `i` and the contents of `outer_list` in this program at **POINT 1**? Describe your invariant in English.

What can we say about `outer_list` at **POINT 2**? Describe your invariant in English.

Note: To answer these questions, see the discussion of this problem on the next page.

Discussion: The issue is that, because we're considering the invariant at the top of the loop, we're really looking at the state of the program at the end of the previous iteration --- except that the value of the loop index `i` has been updated.

- So what we need to do is consider the value of the loop index at the end of the previous iteration, i.e., `i-1`. The invariant would then be a statement about `outer_list[i-1]`.
- But this introduces a boundary-condition problem: at the very beginning of the loop, when `i == 0`, the expression `outer_list [i-1]` refers to the last element of the list.
- So you must consider two possibilities: `i == 0`, which gives one invariant corresponding to what must be true right at the beginning of the loop before it has iterated at all; and `i > 0`, which gives another invariant corresponding to the case where the loop has begun iterating (this is the case where we consider `outer_list[i-1]`). One or the other of these two possibilities must occur, and so the two invariants should be combined using an **or** operator.

To help you formulate the invariants, consider following these steps:

1. Describe what the inner loop does.

2. Describe what the outer loop does.

3. At Point 1, what can we say here, expressed in terms of i?
   How many times has the outer loop iterated?
   How many elements does outer_list have at that point?
   What is the element of outer_list at position i-1?
   What is the length of that list?
   With these answers, write the invariant for POINT 1.

4. Given the description you wrote in 3 above, what can you say at Point 2 expressed in terms of n?