CSc 120: Introduction to Computer Programming II

ICA-5 Solutions

Problem 1

The point of this problem is to illustrate that we can reason about invariants even for dictionary code, where the order in which the keys are processed is unpredictable.

On each iteration, the loop takes one key and concatenates that at the end of the variable `string`. As discussed in class, the invariant at the beginning of iteration i is what must be true at the of iteration i-1, and this is the accumulated effect of the first i-1 iterations. In this case we can say that at the top of the loop, the invariant is that the variable `string` consists of i-1 keys concatenated together in some (unpredictable) order.

At the end of the loop (just before the **return**), the loop invariant is then that the variable `string` consists of all the keys in the dictionary concatenated together in some (unpredictable) order.

We can (and should) check that this is correct even when the dictionary is empty and the **for** loop is not entered.

Problem 2

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

1. Describe what the inner loop does.
   (Answer: it creates a list of length n where each element is n.)
2. Describe what the outer loop does.
   (Answer: The outer loop appends each list created above onto outer_list.)
3. At Point 1, what can we say here, expressed in terms of i?
   How many times has the outer loop iterated?
   (Answer: i times.)
   How many elements does outer_list have at that point?
   (Answer: i)
   What is the element of outer_list at position i-1?
   Answer: if i > 0, then [n,n,n, …, n])
   What is the length of that list?
   (Answer: n)
   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?

a) Invariant at POINT 1:

The inner for loop creates a list of length n containing n elements each of whose values is n (i.e., inner_list = [n, n, n, …, n] of length n). The outer for loop appends each such list into outer_list.

At POINT 1, the outer for loop has iterated i times. So the invariant is that `outer_list` contains i elements each of which is a list of length n with each element having the value n.

To state this with an OR, we could say

i == 0 and `outer_list` is empty
OR
i > 0 and `outer_list` contains i elements, each of which is a list of length n with each element having the value n.

b) Invariant at POINT 2:
At POINT 2, `outer_list` is a list of n lists, each of length n (and containing n occurrences of the number n).