CPSC 331 — Solutions for Tutorial Exercise #3

Questions About Inline Documentation

1. You were first asked to define an ***assertion*** and to explain why it can be helpful to include assertions as inline documentation in code.

   Solution: An ***assertion*** is...

   - a *boolean condition* (that is, "*predicate*")...
   - ...involving an algorithm's *inputs*, *local variables*, *outputs*, and (possibly) *global data*...
   - ...that is assumed to be satisfied *at a specific point in the computation*, namely, either immediately before the execution of the algorithm begins, or immediately before or after a specific instruction in the algorithm has been executed — assuming that the problem's precondition was satisfied when this execution of the algorithm began.

You were next asked to consider the "Fibonacci Number Computation" problem introduced in Lecture #2, and the `fib` algorithm that solves this problem. This algorithm is shown in Figure 1 on page 2. Assertions were given for various places in this algorithm during that lecture.

2. You were asked to say whether each of the following is an ***assertion*** that could be listed after step 5. If the answer is "no", then you were to explain why it is not an assertion. If the answer is "yes", then you were to explain why this is (arguably) inferior to the assertion given, for this point in the algorithm, during Lecture #2.

   (a) ***Assertion:*** This algorithm applies the recursive definition to compute the $\mathtt{n}^{\text{th}}$ Fibonacci number on input $\mathtt{n}$.

   No: This is not an assertion. While it does provide some useful information about the algorithm, it does not describe the state of the computation when a specific point has been reached.

```
integer fib (integer n) {
1. if (n == 0) {
2.   return 0
3. } else if (n == 1) {
4.   return 1
   } else {
5.   return fib(n−2) + fib(n−1)
   }
}
```

Figure 1: An Algorithm for the “Fibonacci Number Computation” Problem

(b) ***Assertion:***

1. $\mathtt{n}$ is an integer input such that $\mathtt{n} \geq 2$.
2. The value $\mathtt{fib}(\mathtt{n}-2)+\mathtt{fib}(\mathtt{n}-1)$ has been returned as output.

Yes, this an assertion, but it is not a very good one. Recall that assertions should be used to explain why an algorithm is correct. Sometimes they can even be written so that they are checked (when “assertions are enabled”) in order to make code “self-testing”.

While the assertion included as inline document for the Python and Java implementations does include the above first point, it continues with a second point that is somewhat more informative:

“The $\mathtt{n}^{\text{th}}$ Fibonacci number, $F_{\mathtt{n}} = F_{\mathtt{n}-2} + F_{\mathtt{n}-1}$, has been returned as output.”

This includes more information about the proof of correctness of this algorithm, as given in Lecture #2. That is, it gives more information about *why* this algorithm is correct.

Questions about Proofs of Correctness

The remaining questions concern a computational problem that is related to the “Fibonacci Number Computation” problem discussed in Lecture #2:

Fibonacci Pair Computation

Precondition: A nonnegative integer $\mathtt{n}$ is given as input.

Postcondition: A array with length 2, whose first and second elements are the Fibonacci numbers $F_{\mathtt{n}}$ and $F_{\mathtt{n}+1}$ respectively, is returned as output.

```
integer[] fibPair (integer n) {
// Assertion:  A nonnegative integer n has been given as input
1. integer[] F = new integer[2]
2. if (n == 0) {
3.   F[0] := 0
4.   F[1] := 1
   } else {
5.   integer[] oldF := fibPair($n-1$)
6.   F[0] := oldF[1]
7.   F[1] := oldF[0] + oldF[1]
   }
8. return F
}
```

Figure 2: An Algorithm for the "Fibonacci Pair Computation" Problem

3. You were asked to suppose that an algorithm `fibPair` correctly solves the above problem, to write a *very* short and simple algorithm that uses this as a subroutine to solve the "Fibonacci Number Computation" problem, and to explain why your answer is correct.

   Solution: An algorithm that solves this problem is as follows.

   ```
   integer fib (integer n) {
   1. integer[] F := fibPair(n)
   2. return F[0]
   }
   ```

   Suppose that the above algorithm is executed with a nonnegative integer n as input. It follows by the correctness of the `fibPair` algorithm that the step at line 1 terminates, and that $\mathrm{F}[0] = F_{\mathrm{n}}$ and $\mathrm{F}[1] = F_{\mathrm{n}+1}$. Thus F_{n} is returned as output after the execution of the step at line 2, as required. Since this algorithm has no undocumented side-effects it follows that it correctly solves the "Fibonacci Number Computation" problem.

A recursive algorithm (which does, indeed, solve this problem is as shown in Figure 2, above.

4. You were asked to write down the ***trace(s) of execution*** corresponding to an execution of the above algorithm on input $\mathrm{n} = 3$, as well as the corresponding ***recursion tree***. You were also asked to discuss the *shape* of the recursion tree, and how this differs from the recursion tree for an execution of the `fib` algorithm on the same input.

Solution:

Execution #1: The integer $\mathtt{n} = 3$ is received as input.

1. $\mathtt{F}$ is declared to be an integer array with length 2 at line 1.
2. The test at line 2 is executed. Since $\mathtt{n} = 3 \neq 0$ this test fails, and the execution continues with the step at line 5. This algorithm is called recursively with input $\mathtt{n} - 1 = 2$ — see Execution #2 for details.
3. As a result of this, $\mathtt{oldF}$ is set to be an integer array with length 2 such that $\mathtt{oldF}[0] = F_2 = 1$ and $\mathtt{oldF}[1] = F_3 = 2$.
4. $\mathtt{F}[0]$ is set to be $\mathtt{oldF}[1] = F_3 = 2$ at line 6.
5. $\mathtt{F}[1]$ is set to be $\mathtt{oldF}[0] + \mathtt{oldF}[1] = F_2 + F_3 = F_4 = 3$ at line 7.
6. The array $\mathtt{F}$ is returned as output at line 8.

Execution #2: The integer $\mathtt{n} = 2$ is received as input.

1. $\mathtt{F}$ is declared to be an integer array with length 2 at line 1.
2. The test at line 2 is executed. Since $\mathtt{n} = 2 \neq 0$ this test fails, and the execution continues with the step at line 5. This algorithm is called recursively with input $\mathtt{n} - 1 = 1$ — see Execution #3 for details.
3. As a result of this, $\mathtt{oldF}$ is set to be an integer array with length 2 such that $\mathtt{oldF}[0] = F_1 = 1$ and $\mathtt{oldF}[1] = F_2 = 1$.
4. $\mathtt{F}[0]$ is set to be $\mathtt{oldF}[1] = F_2 = 1$ at line 6.
5. $\mathtt{F}[1]$ is set to be $\mathtt{oldF}[0] + \mathtt{oldF}[1] = F_1 + F_2 = F_3 = 2$ at line 7.
6. The array $\mathtt{F}$ is returned as output at line 8.

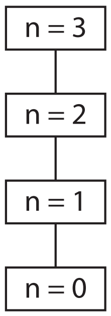
Execution #3: The integer $\mathtt{n} = 1$ is received as input.

1. $\mathtt{F}$ is declared to be an integer array with length 2 at line 1.
2. The test at line 2 is executed. Since $\mathtt{n} = 1 \neq 0$ this test fails, and the execution continues with the step at line 5. This algorithm is called recursively with input $\mathtt{n} - 1 = 0$ — see Execution #4 for details.
3. As a result of this, $\mathtt{oldF}$ is set to be an integer array with length 2 such that $\mathtt{oldF}[0] = F_0 = 0$ and $\mathtt{oldF}[1] = F_1 = 1$.
4. $\mathtt{F}[0]$ is set to be $\mathtt{oldF}[1] = F_1 = 1$ at line 6.
5. $\mathtt{F}[1]$ is set to be $\mathtt{oldF}[0] + \mathtt{oldF}[1] = F_0 + F_1 = F_2 = 1$ at line 7.
6. The array $\mathtt{F}$ is returned as output at line 8.

Execution #4: The integer $\mathtt{n} = 0$ is received as input.

1. $\mathtt{F}$ is declared to be an integer array with length 2 at line 1.
2. The test at line 2 is executed.Since $\mathtt{n} = 0$ this test is passed, so that execution of the algorithm continues with the step at line 3.
3. $\mathtt{F}[0]$ is set to be $F_0 = 0$ at line 3.
4. $\mathtt{F}[1]$ is set to be $F_1 = 1$ at line 4.
5. The array $\mathtt{F}$ is returned as output at line 8.

The corresponding recursion tree is as follows.



It is notable that this is simply simpler and smaller than the recursion tree for the execution of the algorithm in the lecture about this: Rather than being very large and spread out, this is just a single chain.

5. You were asked to write down ***assertions*** for the following points during the execution of this algorithm:

   (a) Immediately after the execution of the step at line 1;
   Solution: *Assertion:*
   1. $\mathtt{n}$ is a non-negative integer input.
   2. $\mathtt{F}$ is an integer array with length two.

   (b) Immediately after the execution of the step at line 4;
   Solution: *Assertion:*
   1. $\mathtt{n}$ is a non-negative integer input such that $\mathtt{n} = 0$.
   2. $\mathtt{F}$ is an integer array with length two such that $\mathtt{F}[0] = F_{\mathtt{n}} = F_0 = 0$ and $\mathtt{F}[1] = \mathrm{F}_{\mathtt{n}+1} = F_1 = 1$.

(c) Immediately *before* the execution of the step at line 5;
Solution: *Assertion:*
1. n is a non-negative integer input such that $\mathtt{n} \geq 1$.
2. F is an integer array with length two.

(d) Immediately after the execution of the step at line 7;
Solution: *Assertion:*
1. n is a non-negative integer input such that $\mathtt{n} \geq 1$.
2. F is an integer array with length two such that $\mathtt{F} = F_{\mathtt{n}}$ and $\mathtt{F}[1] = F_{\mathtt{n}-1} + F_{\mathtt{n}} = F_{\mathtt{n}+1}$.

(e) Immediately *before* the execution of the step at line 8;
Solution: *Assertion:*
1. n is a non-negative integer input.
2. F is an integer array with length two such that $\mathtt{F}[0] = F_{\mathtt{n}}$ and $\mathtt{F}[1] = F_{\mathtt{n}+1}$.

(f) Immediately *after* the execution of the step at line 8.
Solution: *Assertion:*
1. n is a non-negative integer input.
2. An integer array F with length two, such that $\mathtt{F}[0] = F_{\mathtt{n}}$ and $\mathtt{F} = F_{\mathtt{n}+1}$, has been returned as output.

Note: It might be helpful to complete Problem #7 before attempting this one.

6. Write a reasonably short proof that the function $f(\mathtt{n}) = \mathtt{n}$ is a ***bound function*** for this recursive algorithm.

   Solution: Since n is an integer-valued input, this is certainly well-defined total function of this recursive algorithm's inputs.

   This algorithm only calls itself recursively when the step at line 5 is executed — with the input n replaced by the input $\mathtt{n} - 1$, so that the value of the function f is decreased by (at least) one every time this function calls itself recursively.

   If the precondition for the problem is satisfied then n is a non-negative integer input. On the other hand, if $f(\mathtt{n}) = \mathtt{n} \leq 0$ as well then $\mathtt{n} = 0$. Consequently, an execution of the algorithm halts with the execution of the steps at lines 1–4 and 8 — without the algorithm calling itself recursively.

   It follows that the function $f(\mathtt{n}) = \mathtt{n}$ is a bound function for this recursive algorithm, because is satisfies all the properties included in the definition of this.

7. Finally, you were asked to write a proof that this algorithm correctly solves the "Fibonacci Pair Computation" problem.

   Solution: It is reasonably easy to see, by inspection of the code, that the input is not changed and that no global data is accessed or modified — so that it is necessary and sufficient to prove the following:

Claim: If the `fibPair` algorithm is executed with a nonnegative integer $\mathtt{n}$ as input, then this execution of the algorithm eventually ends, and an integer array $\mathtt{F}$ with length two such that $\mathtt{F[0]} = F_{\mathtt{n}}$ and $\mathtt{F[1]} = F_{\mathtt{n}+1}$ is returned as output.

Proof. By induction on $\mathtt{n}$. The standard form of mathematical induction will be used.

Basis: Consider an execution of the `fibPair` algorithm with input $\mathtt{n} = 0$. One can see by inspection of the code that — since $\mathtt{n} = 0$, so that the test at line 2 is passed — this execution of the algorithm includes the steps at lines 1–4 and 8.

One can therefore see, by an examination of the code at these steps, that this execution of the algorithm halts and an integer array $\mathtt{F}$ with length two such that $\mathtt{F}[0] = F_{\mathtt{n}} = F_0 = 0$ and $\mathtt{F}[1] = F_{\mathtt{n}+1} = F_1 = 1$ is returned as output, as required to establish the claim in this case.

Inductive Step: Let k be an integer such that $k \geq 0$. It is necessary and sufficient to use the following

> Inductive Hypothesis: If the `fibPair` algorithm is executed with $\mathtt{n} = k$ as input, then this execution of the algorithm eventually ends, and an integer array $\mathtt{F}$ with length two such that $\mathtt{F}[0] = F_{\mathtt{n}} = F_k$ and $\mathtt{F}[1] = F_{\mathtt{n}+1} = F_{k+1}$ is returned as output.

to prove the following

> Inductive Claim: If the `fibPair` algorithm is executed with $\mathtt{n} = k+1$ as input, then this execution of the algorithm eventually ends, and an integer array $\mathtt{F}$ with length two such that $\mathtt{F}[0] = F_{\mathtt{n}} = F_{k+1}$ and $\mathtt{F}[1] = F_{\mathtt{n}+1} = F_{k+2}$ is returned as output.

With that noted, consider an execution of the `fibPair` algorithm with the integer $\mathtt{n} = k+1$ given as input. Since $k \geq 0$, $\mathtt{n} = k+1 \geq 1$, so that the execution of the algorithm includes the steps at lines 1, 2, and 5–8.

Since $\mathtt{n} - 1 = k$, it follows by the Inductive Hypothesis that the recursive execution of the `fibPair` algorithm at line 5 eventually halts, and that $\mathtt{F}$ is set to be an integer array with length two such that $\mathtt{oldF}[0] = F_k$ and $\mathtt{oldF}[1] = F_{k+1}$.

$\mathtt{F}[0]$ is therefore set to be $F_{\mathtt{n}} = F_{k+1}$ at line 6 and, since $\mathtt{n}+1 = k+2 \geq 2$, $\mathtt{oldF}[1]$ is set to be $F_{\mathtt{n}+1} = F_k + F_{k+1} = F_{k+2}$ at line 7. The execution of the algorithm then ends after the execution of line 8 — establishing the inductive claim, as needed to complete the inductive step and the proof. $\square$