CPSC 331 — Solutions for Tutorial Exercise #6

The initial questions on this exercised concerned the `betterFibLoop` algorithm, which also solves the "Fibonacci Number Computation" problem, and which is shown in Figure **??** on page **??**.

1. You were first asked to show that $\mathtt{n} - \mathtt{i}$ is a ***bound function*** for the `while` loop in this algorithm.

   Solution: See the solution for Problem #3 in Tutorial Exercise #4: This bound function was identified, and proved to be correct, there.

2. You were next asked to use this to provide an upper bound for the number of steps executed by this algorithm as part of its execution on a non-negative integer input `n`.

   Solution: If $\mathtt{n} = 0$ then the steps at lines 1 and 2 are executed and the algorithm halts. Since the "Uniform Cost Criterion" is being used here the number of steps executed (and "running time") is 2 in this case.

   Suppose, instead, that $\mathtt{n} \geq 1$.

   - The test at line 1 is checked and failed, so the steps at lines 1, and 3–5 are executed before step 6 — so 4 steps are executed before the `while` loop is reached.
   - The initial value for the bound function for this loop (identified when answering the previous question) is $\mathtt{n} - 1$, so there are at most $\mathtt{n} - 1$ executions of the body of the `while` loop and at most `n` executions of the loop test.
   - The loop body just consists of four statements, at lines 7–10. There are no loops or tests here and no other methods are called, so every execution of the loop body includes 4 steps: For $1 \leq j \leq \mathtt{n} - 1$, the cost of the j^{th} execution of the loop body is
     $$T_{\text{body}}(j) = 4,$$
     if there is actually a j^{th} execution of the body of the loop at all.
   - Similarly, the loop test just consists of a single statement which does not call any other methods — so, for $1 \leq j \leq \mathtt{n}$, the cost of the j^{th} execution of the loop test is
     $$T_{\text{test}}(j) = 1,$$

```
integer betterFibLoop (integer n) {
// Assertion:  A nonnegative integer n has been given as input.
 1. if (n == 0) {
 2.  return 0
   } else {
 3.  integer oldest := 0
 4.  integer middle := 1
 5.  integer i := 1
 6.  while (i < n) {
 7.   integer youngest := oldest + middle
 8.   oldest := middle
 9.   middle := youngest
10.   i := i + 1
     }
11.  return middle
   }
}
```

Figure 1: The `betterFibLoop` Algorithm

if a j^{th} execution of the loop test is ever performed.

- It now follows the total number of steps included in this execution of the loop is at most

$$\begin{aligned}\sum_{j=1}^{\mathtt{n}-1} T_{\text{body}}(j) + \sum_{j=1}^{\mathtt{n}} T_{\text{test}}(j) &= \sum_{j=1}^{\mathtt{n}-1} 4 + \sum_{j=1}^{\mathtt{n}} 1 \\ &= 4(\mathtt{n}-1) + \mathtt{n} \\ &= 5\mathtt{n} - 4.\end{aligned}$$

- There is only one more statement executed after this — at line 11 — and the cost of this is 1.
- Since $4 + (5\mathtt{n} - 4) + 1 = 5\mathtt{n} + 1$, it now follows that the number of steps included in an execution of this algorithm on a non-negative integer input $\mathtt{n}$ is

$$T(n) = \begin{cases} 2 & \text{if } \mathtt{n} = 0, \\ 5\mathtt{n} + 1 & \text{if } \mathtt{n} \geq 1. \end{cases}$$

The remaining questions concerned the `fibPair` algorithm, introduced in Tutorial Exercise #3, which solves the "Fibonacci Pair Computation" problem. This is as shown in Figure **??**.

```
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 write a ***recurrence*** for the number $T(\mathtt{n})$ of numbered steps executed by this algorithm on a non-negative integer input $\mathtt{n}$.

   Solution: If the algorithm is executed when $\mathtt{n} = 0$ then 5 numbered steps are executed — namely, the steps at lines 1, 2, 3, 4, and 8.

   If the algorithm is executed with $\mathtt{n} \geq 1$ then 6 numbered steps — at lines 1, 2, 5, 6, 7 and 8 are executed. However, the step at line 5 also includes a ***recursive application*** of this algorithm with input $\mathtt{n} - 1$.

   Thus the number, $T(\mathtt{n})$, of number of steps executed by this algorithm on a non-negative integer input $\mathtt{n}$ is given by the following recurrence:

$$T(\mathtt{n}) = \begin{cases} 5 & \text{if } \mathtt{n} = 0, \\ T(\mathtt{n}-1) + 6 & \text{if } \mathtt{n} \geq 1. \end{cases}$$

   Non-Solutions: *Neither* of the answers

$$T(\mathtt{n}) = \begin{cases} 5 & \text{if } \mathtt{n} = 0, \\ 6\mathtt{n} + 5 & \text{if } \mathtt{n} \geq 1. \end{cases}$$

   or

$$T(\mathtt{n}) = 6\mathtt{n} + 5$$

   would be an acceptable on a test or assignment in this course.

   Why These Answers are Not Acceptable: They are not immediately verifiable from the given source code and also suggest that you do not know what a ***recurrence*** is.

Furthermore, they might make a later answer — "Solve the Recurrence" — impossible, because you would not know what to do.

4. You were next asked to guess a simple ***solution*** for this recurrence.

   Solution: $T(\mathtt{n}) = 6\mathtt{n} + 5$.

   How This Could Be Guessed! Look for a pattern! By definition, $T(\mathtt{n}) - T(\mathtt{n}-1) = 6$ for every integer $\mathtt{n} \geq 1$. This may suggest that $T(\mathtt{n}) = 6\mathtt{n} + c$ for some constant c. Checking $T(0)$ confirms that $c = T(0) = 5$.

5. Finally, you were asked to use your recurrence to prove that your solution is correct.

   Solution: It is necessary to state and prove the following:

   Claim: If T is a function of the non-negative integers such that, for every non-negative integer $\mathtt{n}$,

$$T(\mathtt{n}) = \begin{cases} 5 & \text{if } \mathtt{n} = 0, \\ T(\mathtt{n}-1) + 6 & \text{if } \mathtt{n} \geq 1, \end{cases}$$

   then $T(\mathtt{n}) = 6\mathtt{n} + 5$ for every integer $\mathtt{n} \geq 0$.

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

   Basis: If $\mathtt{n} = 0$ then $T(\mathtt{n}) = T(0) = 5$, by definition, while $6\mathtt{n} + 5 = 6 \cdot 0 + 5 = 5$ as well, as needed 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: $T(k) = 6k + 5$.

   to prove the following

   Inductive Claim: $T(k+1) = 6(k+1) + 5$.

   Since $k \geq 0$, $k + 1 \geq 1$, and it follows by the recursive definition of T that

$$\begin{aligned} T(k+1) &= T((k+1)-1) + 6 \\ &= T(k) + 6 \\ &= (6k+5) + 6 && \text{(by the Inductive Hypothesis)} \\ &= 6(k+1) + 5 \end{aligned}$$

   as needed to establish the Inductive Claim, complete the Inductive Step, and establish the claim. □