Math 15 - Spring 2017 - Homework 5.5 Solutions

1. (5.5 # 11) Prove that the following algorithm is correct.
   `Preconditions:` $n = 2^p$ for some integer $p \geq 0$.
   `Postconditions:` `PLog`$(n) = log_2 n$.
   `function PLog`$(n \in \mathbb{N})$
   `if` $n = 1$ `then`
   `return 0`
   `else`
   `return` 1+`PLog`$(n/2)$
   Proof (by induction on p):
   Base case: If $p = 0$, we have `PLog`$(1) = 0 = \log_2 1$.
   Inductive hypothesis: `PLog`$(2^k - 1) = log_2(2^k - 1) = k - 1$, for some $k > 0$. Then

$$\begin{aligned} \texttt{PLog}(2^k) &= 1 + \texttt{PLog}(2^{k-1})\texttt{, by definition} \\ &= 1 + (k-1)\texttt{, by the inductive hypotheses} \\ &= k\texttt{, hooray!} \end{aligned}$$

2. (5.5 # 12) Prove that the following algorithm is correct.
   `Preconditions:` $n \geq 1$.
   `Postconditions:` `ILog`$(n) = \lfloor log_2 n \rfloor$.
   `function ILog`$(n \in \mathbb{N})$
   `if` $n = 1$ `then`
   `return 0`
   `else`
   `return` 1+`ILog`$(\lfloor n/2 \rfloor)$
   Proof (by induction on n):
   Base case: If $n = 1$, we have `ILog`$(1) = 0 = \log_2 1$.
   (Strong) inductive hypothesis: `ILog`$(k) = \lfloor log_2(k) \rfloor$, for $0 \leq k < m - 1$, for some $m > 1$. Let p be the greatest power of 2 less than or equal to $m - 1$, so $\lfloor log_2(m-1) \rfloor = p$.

   Since $2^p \leq m - 1$, $2^{p-1} \leq (m-1)/2$, and since $2^{p-1} \in \mathbb{Z}^+$, $2^{p-1} \leq \lfloor (m-1)/2 \rfloor$.

   Also, $2^p > (m-1)/2 \geq \lfloor (m-1)/2 \rfloor$, since p is the greatest power of two less than or equal to $m - 1$. Thus 2^{p-1} is the greatest power of two less than or equal to $\lfloor (m-1)/2 \rfloor$. By inductive hypothesis, `ILog`$(\lfloor (m-1)/2 \rfloor) = \lfloor (log_2 \lfloor (m-1)/2 \rfloor) \rfloor = p - 1$. By definition `ILog`$(m-1) =$ 1+`ILog`$(\lfloor (m-1)/2 \rfloor) = 1 + p - 1 = p$, hooray...

3. (5.5 # 18) Prove that the following recursive sequential search algorithm is correct.
   `Algorithm 5.20 Recursive Sequential Search.`
   `Preconditions: The set` U `is totally ordered by` $<$, `and` $X = \{x_1, x_2, \ldots, x_n\}$ is a (possibly empty) subset of U.
   `Postconditions:` `RSearch` $(t, X) = (t \in X)$.
   `function RSearch` $(t \in U, X \subset U)$
   `if` $X = \emptyset$ `then`
   `return false`
   `else`
   `return` $(t = x_n) \vee$`RSearch` $(X \setminus \{x_n\})$
   ANS: Proof (by induction on n, the size of X.)
   If $n = 0$, then $X = \emptyset$, which does not contain t, and so `RSearch`(t, X) returns false.
   The inductive hypothesis assumes that `RSearch`(t, X) returns true if and only if $t \in X$, for $|X| = k - 1$,

for some $k > 0$.
Let X be a set with k elements. Then $\texttt{RSearch}(t, X)$ returns $(t = x_k) \vee \texttt{RSearch}(t, X \setminus \{x_k\})$.
By the inductive hypothesis, $\texttt{RSearch}(t, X \setminus \{x_k\})$ will be true if and only if $t \in X \setminus \{x_k\}$. Therefore, $t \in X \Leftrightarrow (t = x_k) \vee (t \in X \setminus \{x_k\}) \Leftrightarrow (t = x_k) \vee \texttt{RSearch}(t, X \setminus \{x_k\}) \Leftrightarrow \texttt{RSearch}(t, X)$, as desired.

4. (5.5 # 20) Prove: If $x|m$ and $x|(n \mod m)$, then $x|n$.
ANS: Proof. Suppose $x|m$ and $x|(n \mod m)$. Then by definition there is some integer k such that $m = kx$ and there is some integer j such that $r = jx$, where $r = n \mod m$. By the definition of the mod operation, there is some integer q such that $n = qm + r$. Therefore $n = qkx + jx = (qk + j)x$, so $x|n$.

5. (5.5 # 22) Test Algorithm 5.19 for solving the Towers of Hanoi puzzle in the case of $n = 3$ disks using a top-down evaluation.
ANS:

$$\begin{aligned}\texttt{H}(3,1,3) &= \texttt{H}(2,1,2),(1,3),\texttt{H}(2,2,3)\\ &= \texttt{H}(1,1,3),(1,2),\texttt{H}(1,3,2),(1,3),\texttt{H}(1,2,1),(2,3),\texttt{H}(1,1,3)\\ &= (1,3),(1,2),(3,2),(1,3),(2,1),(2,3),(1,3)\end{aligned}$$

Here's the Python code we produced in class:

```
def Hanoi(n,start,goal):
    if n==1:
        return (start,goal)
    else:
        temp = 6-(start+goal)
        return Hanoi(n-1,start,temp),(start,goal),Hanoi(n-1,temp,goal)
```

If you then `print(Hanoi(3,1,3))`, the output is

```
(((1, 3), (1, 2), (3, 2)), (1, 3), ((2, 1), (2, 3), (1, 3)))
```

The extra parentheses add some clarity: First move the top two discs from 1 to 2, then move the bottom disc from 1 to 3, then move the two discs from 2 to 3, using 1 as the temp.
If you `print(Hanoi(8,1,3))`, the output is longer, but has the same basic flavor:

```
(((((((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2))),
  (1, 3),
     (((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3)))),
  (1, 2),
    ((((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1))),
  (3, 2),
     (((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2)))),
  (1, 3),
   (((((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3))),
  (2, 1),
     (((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1)))),
  (2, 3),
    ((((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2))),
  (1, 3),
     (((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3)))))),
  (1, 2),
```

```
((((((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1))),
(3, 2),
   (((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2)))),
(3, 1),
  ((((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3))),
(2, 1),
   (((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1)))))),
(3, 2),
 (((((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2))),
(1, 3),
   (((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3)))),
(1, 2),
  ((((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1))),
(3, 2),
   (((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2)))))))),
(1, 3),
(((((((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3))),
(2, 1),
   (((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1)))),
(2, 3),
  ((((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2))),
(1, 3),
   (((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3)))))),
(2, 1),
 (((((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1))),
(3, 2),
   (((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2)))),
(3, 1),
  ((((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3))),
(2, 1),
   (((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1))))))),
(2, 3),
((((((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2))),
(1, 3),
   (((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3)))),
(1, 2),
  ((((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1))),
(3, 2),
   (((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2)))))),
(1, 3),
 (((((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3))),
(2, 1),
   (((3, 1), (3, 2), (1, 2)), (3, 1), ((2, 3), (2, 1), (3, 1)))),
(2, 3),
  ((((1, 2), (1, 3), (2, 3)), (1, 2), ((3, 1), (3, 2), (1, 2))),
(1, 3),
   (((2, 3), (2, 1), (3, 1)), (2, 3), ((1, 2), (1, 3), (2, 3)))))))
```

6. (5.5 # 24) Use a recurrence relation to show that Algorithm 5.19 (the Towers of Hanoi algorithm) will always return a sequence of $2^n - 1$ ordered pairs.
   ANS: First we want a recurrence relation. Let $P(n) =$ the number of pairs returned by `Hanoi`(n, T_1, T_2).

If there's only one disc, then `Hanoi` will return one pair `(start,goal)`, so `Hanoi(start,goal) = (start,goal)`. If $n > 1$, then we have the recursion $P(n) = P(n-1) + 1 + P(n1)$, given $2P(n-1)+1$ pairs. The recurrence relation is,

$$P(n) = \begin{cases} 1 & : n = 1 \\ 2P(n-1) & : n > 1 \end{cases}$$

We are to prove that $P(n) = 2^n - 1$, which we do in the usual manner of induction on n. If $n = 1$, $P(1) = 1 = 2^1 - 1$. The inductive hypothesis has that $P(k-1) = 2^{k-1} - 1$, for $k > 1$.
Now $P(k) = 2P(k-1) + 1 = 2 \cdot (2^{k1} - 1) + 1 = 2^k - 1$, as desired.