7. There is nothing to do if $n = 0$.

```
procedure simpler_hanoi(X, Y, Z : peg names, n : natural number)
    {assume that the peg names are A, B, and C in some order}
    if n > 0 then
        begin
            call simpler_hanoi(X, Z, Y, n − 1)
            print("Move disk " n " from peg " X " to peg " Y ".")
            call simpler_hanoi(Z, Y, X, n − 1)
        end
    return
```

9. **(a)** We need to compare l_n to the largest of the first $n-1$ elements in the list.

```
procedure largest(L : list of numbers)
    {assume that L = (l_1, l_2, ..., l_n), with n ≥ 1}
    if n = 1 then return(l_1)
        else
            begin
                a ← largest((l_1, l_2, ..., l_{n−1}))
                if l_n > a then return(l_n)
                    else return(a)
            end
```

(b) This is the same part (a) except that the procedure returns the index rather than the value.

```
procedure largest_index(L : list of numbers)
    {assume that L = (l_1, l_2, ..., l_n), with n ≥ 1}
    if n = 1 then return(1)
        else
            begin
                i ← largest_index((l_1, l_2, ..., l_{n−1}))
                if l_n > l_i then return(n)
                    else return(i)
            end
```

11. **(a)** We just copy the definition into the recursive procedure.

```
procedure fib_3(n : positive integer)
    if n ≤ 3 then return(1)
        else return(fib_3(n − 1) + fib_3(n − 2) + fib_3(n − 3))
```

(b) This is exactly analogous to the iterative algorithm for calculating the terms of the Fibonacci sequence.

```
procedure fib_3_iterative(n : positive integer)
    x ← 1; y ← 1; z ← 1
    for i ← 4 to n do
      begin
        next ← x + y + z
        x ← y; y ← z; z ← next
      end
    return(z)
```

(c) The number of steps is in $O(n)$, since the loop is executed $O(n)$ times.

13.

```
procedure valid(s : string)
  {assume that s = s_1 s_2 ... s_n, with n ≥ 1}
    if n = 1 and s_i is a letter then return(true)
      else if s_n is a letter or digit then return(valid(s_1 s_2 ... s_{n-1}))
        else return(false)
```

15. Note how each procedure recursively calls the other, as well as itself.

```
procedure F(n : natural number)
    if n = 0 then return(1)
      else return(n − M(F(n − 1)))
procedure M(n : natural number)
    if n = 0 then return(0)
      else return(n − F(M(n − 1)))
```

17. In effect this procedure searches from back to front.

```
procedure linear(A, x)
  {assume the setting of Algorithm 1 in Section 4.3}
    if n = 0 then return(0)
      else if x = a_n then return(n)
        else return(linear((a_1, a_2, ..., a_{n-1}), x))
```

19. $O(n+m)$, since the loop statement is executed $n+m$ times

21. (a)

```
procedure element_of(x : number, B : list of numbers)
  {assume that B = (b_1, b_2, ..., b_n), with n ≥ 0}
    if n = 0 then return(false) {x is not in the empty list}
      else if x = b_n then return(true) {x is there}
        else return(element_of(x, (b_1, b_2, ..., b_{n-1})))
          {look for x in the rest of the list}
```

(b)

```
procedure subset_of(A, B : lists of numbers)
  {assume that A = (a_1, a_2, ..., a_m), with m ≥ 0}
    if m = 0 then return(true) {∅ ⊆ B always holds}
      else return(element_of(a_m, B) ∧ subset_of((a_1, a_2, ..., a_{m-1}), B))
        {see if a_m ∈ B and the first m − 1 elements are all in B}
```

(c) **procedure** *equals*(A, B : lists of numbers)
return(*subset_of*(A, B) $\wedge$ *subset_of*(B, A))

(d) **procedure** *adjoin*(x : number, B : list of numbers)
{assume that $B = (b_1, b_2, \ldots, b_n)$, with $n \geq 0$}
return($(b_1, b_2, \ldots, b_n, x)$)

(e) **procedure** *reduce*(A : list of numbers)
{assume that $A = (a_1, a_2, \ldots, a_n)$, with $n \geq 0$}
if $n = 0$ **then return**(A) {the empty set}
else if *element_of*($a_n, (a_1, a_2, \ldots, a_{n-1})$)
then return(*reduce*($(a_1, a_2, \ldots, a_{n-1})$))
{a_n is superfluous}
else return(*adjoin*(a_n, *reduce*($(a_1, a_2, \ldots, a_{n-1})$)))
{reduce the rest and then put a_n back in}

23. The idea here is to list all the subsets that do not include i, and then those that do, as i recursively ranges from n down to 1. The string s holds the final portion of the subset, involving numbers greater than i. Recall that λ is the empty string.

```
procedure subsets(n : positive integer)
    {prints all subsets of {1, 2, 3, ..., n}}
        call rec_subsets(n, n, λ)
        return
procedure rec_subsets(n : positive integer, i : natural number, s : string)
    {prints all subsets of {1, 2, 3, ..., n} that consist of the elements listed
     in s and any subset of {1, 2, 3, ..., i}}
      if i = 0 then print(s)
          else
              begin
                  call rec_subsets(n, i − 1, s)
                  s ← the number i followed by a blank, followed by s
                  call rec_subsets(n, i − 1, s)
              end
      return
```

25.

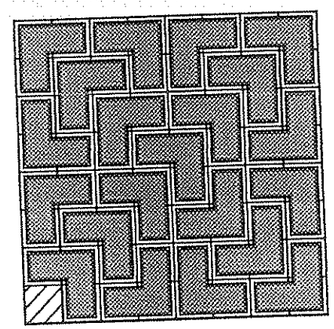
```
procedure EVAL(L : list of items forming a valid FPAE)
    {assume that L = (l_1, l_2, ..., l_n), with n ≥ 1; each l_i is either an unsigned
     integer, a variable, an operator symbol, or a left or right parenthesis}
      if n = 1 then return(EVAL_atom(l_1))
          else
              begin {need to find middle operator symbol}
                  count ← 1
                  for i ← 2 to n − 1 do
                      if count = 1 and l_i is an operator symbol θ then
                              return(EVAL((l_2, ..., l_{i−1})) θ EVAL((l_{i+1}, ..., l_{n−1})))
                          else if l_i = "(" then count ← count + 1
                              else if l_i = ")" then count ← count − 1
              end
```

27. **(a)** It computes an approximation for the so-called arithmetic-geometric mean of 1 and 10. It returns approximately 4.250407.

(b) **procedure** *gauss_recursive*$(x, y, \epsilon$: positive real numbers with $x \le y)$
if $y - x \le \epsilon$ **then return**(x)
else return$((gauss(\sqrt{xy}, (x+y)/2, \epsilon))$

SECTION 5.3 Proof by Mathematical Induction

1. This figure is produced by using the proof given in Example 1. The middle L-shaped piece is placed first, dividing the problem into four smaller problems.



3. The base case ($n = 2$) checks: $1 \cdot 2 = 1 \cdot 2 \cdot 3/3$. Assume the inductive hypothesis, that $1 \cdot 2 + 2 \cdot 3 + \cdots + (k-1)k = [(k-1)k(k+1)]/3$. Then

$$\begin{aligned} 1 \cdot 2 + 2 \cdot 3 + \cdots + (k-1)k + k(k+1) &= \frac{(k-1)k(k+1)}{3} + k(k+1) \\ &= k(k+1)\left(\frac{k-1}{3} + 1\right) = k(k+1)\left(\frac{k+2}{3}\right) \\ &= \frac{((k+1)-1)(k+1)((k+1)+1)}{3}, \end{aligned}$$

as desired.

5. $(S(1) \wedge \forall k \ge 1 : (S(k) \to S(k+1))) \to \forall n : S(n)$

7. The base case ($n = 1$) checks: $1/2 \geq 1/2$. Assume the inductive hypothesis, that

$$\frac{1}{2} \cdot \frac{3}{4} \cdot \frac{5}{6} \cdots \frac{2k-1}{2k} \geq \frac{1}{2k}.$$

Then

$$\begin{aligned}\frac{1}{2} \cdot \frac{3}{4} \cdot \frac{5}{6} \cdots \frac{2k-1}{2k} \cdot \frac{2k+1}{2k+2} &\geq \frac{1}{2k} \cdot \frac{2k+1}{2k+2} \\ &= \frac{2k+1}{2k} \cdot \frac{1}{2k+2} \\ &\geq \frac{1}{2k+2} = \frac{1}{2(k+1)},\end{aligned}$$

as desired.

9. The base case checks: $1 \cdot 1! = 2! - 1$. Assuming the inductive hypothesis, we have

$$\begin{aligned}\sum_{j=1}^{k+1} j \cdot j! &= \sum_{j=1}^{k} j \cdot j! + (k+1)(k+1)! \\ &= (k+1)! - 1 + (k+1)(k+1)! \\ &= (k+1)!(1 + (k+1)) - 1 \\ &= (k+2)(k+1)! - 1 = (k+2)! - 1,\end{aligned}$$

as desired.

11. There are two base cases. For $n = 0$ or $n = 1$, the algorithm uses no additions, and $f(n) - 1 = 1 - 1 = 0$. Assume the inductive hypothesis, that Algorithm 3 requires $f(i) - 1$ addition operations to compute $f(i)$ for all $i < k$, where $k \geq 2$. Then to compute $f(k)$, the algorithm recursively computes $f(k-1)$ and $f(k-2)$ and adds the answers. This requires $[f(k-1) - 1] + [f(k-2) - 1] + 1$ additions (the two bracketed quantities come from the inductive hypothesis). But this is $f(k-1) + f(k-2) - 1$, which equals $f(k) - 1$ by the definition of the Fibonacci sequence, exactly as desired.

13. We experiment a little to find the terms of the sequence whose nth term is

$$\frac{1}{1 \cdot 3} + \frac{1}{3 \cdot 5} + \cdots + \frac{1}{(2n-1)(2n+1)}.$$

The successive values are $1/3$, $2/5$, $3/7$, $4/9$, ... This suggest that the sum is $n/(2n+1)$ in general. The base case has just been verified. Assuming the inductive hypothesis

$$\frac{1}{1 \cdot 3} + \frac{1}{3 \cdot 5} + \cdots + \frac{1}{(2n-1)(2n+1)} = \frac{n}{2n+1}$$

(we are letting n, rather than k, play the role of the variable to induct on now), we compute

$$\begin{aligned}\frac{1}{1\cdot 3}+\cdots+\frac{1}{(2n-1)(2n+1)}+\frac{1}{(2n+1)(2n+3)} &= \frac{n}{2n+1}+\frac{1}{(2n+1)(2n+3)}\\ &= \frac{2n^2+3n+1}{(2n+1)(2n+3)}\\ &= \frac{(2n+1)(n+1)}{(2n+1)(2n+3)}\\ &= \frac{n+1}{2n+3}=\frac{n+1}{2(n+1)+1},\end{aligned}$$

as desired. Thus the given sum (where $n = 50000$) is $50000/100001 \approx 0.499995$.

15. We first see what happens for small values of n: $2^0 = 1 > 0 = 0^2$, $2^1 = 2 > 1 = 1^2$, $2^2 = 4 \not> 4 = 2^2$, $2^3 = 8 \not> 9 = 3^2$, $2^4 = 16 \not> 16 = 4^2$, $2^5 = 32 > 25 = 5^2$, $2^6 = 64 > 36 = 6^2$. We claim that the inequality holds for $n = 0$, $n = 1$, and $n \geq 5$. It remains to prove the last of these claims by mathematical induction, the base case $n = 5$ having already been done. Assume the inductive hypothesis that $2^k > k^2$. Then

$$\begin{aligned}(k+1)^2 &= k^2+(2k+1)\\ &< k^2+k^2 \qquad (\text{since } k\geq 5)\\ &= 2k^2 < 2\cdot 2^k = 2^{k+1}.\end{aligned}$$

17. **(a)** The base case was not checked. In fact it is false: $1 \neq (1+1+4)/2$.

(b) We need to show for *all* $k \geq 1$ that if $2k+1$ is prime then $2(k+1)+1$ is prime, not just this implication for $k = 1$. If fails for $k = 3$, since 7 is prime, but 9 is not.

(c) It is not true that $a_2 \in B$ if $k = 2$. Indeed, in that case, $B = \{a_1\}$, and $C = \{a_2\}$, and a_1 and a_2 need not have the same parity.

(d) Since we used the inductive hypothesis for $i = k-2$, we needed to check two base cases, $n = 0$ and $n = 1$. In fact the proposition fails for $n = 1$ (for any $a \neq 1$).

(e) The base case is *not* true. If $n = 1$, then $1^1 + 1 = 2$ is not odd.

19. Suppose that $S(n)$ is a proposition with a free variable n that ranges over the positive integers. Then if $S(1)$ is true, and if for every $k > 1$ the implication $S(k-1) \to S(k)$ is true, then $S(n)$ is true for all n.

21. Suppose that $S(n)$ is a proposition with a free variable n that ranges over the natural numbers. Then if $S(0)$, $S(1)$, and $S(2)$ are true, and if for every $k > 2$ the implication $(\forall i{<}k\colon S(i)) \to S(k)$ is true, then $S(n)$ is true for all n.

23. By Example 6, $f_{1000} > 1.5^{1000}$. Thus $\log_{10} f_{1000} > 1000 \log_{10} 1.5 \approx 176.09$. Therefore $f_{1000} > 10^{176}$, so f_{1000} has more than 176 decimal digits.

25. We prove that for all $n \geq 0$, if x is a FPAE born on day n, then $LP(x) = RP(x)$. If $n = 0$ (base case), then x must be an unsigned integer or variable name, so $LP(x) = 0 = RP(x)$. Let $k > 0$, and assume the inductive hypothesis, that for all $i < k$, every FPAE y born on day i satisfies $LP(y) = RP(y)$. Let x be a FPAE born on day k; we must show that $LP(x) = RP(x)$. By the recursive definition of FPAE, we must have $x = (\alpha + \beta)$, $x = (\alpha - \beta)$, $x = (\alpha * \beta)$, $x = (\alpha/\beta)$, or $x = (-\alpha)$ for some α and β born prior to day k. The cases are similar, so we will assume that $x = (\alpha * \beta)$. By the inductive hypothesis, $LP(\alpha) = RP(\alpha)$, and $LP(\beta) = RP(\beta)$. But then $LP(x) = 1 + LP(\alpha) + LP(\beta) = 1 + RP(\alpha) + RP(\beta) = RP(x)$, as desired.

27. We want to argue that if the base case and inductive step have been proved, then $S(n)$ must be true for all n. We will argue by contradiction. If $\forall n{:}\, S(n)$ is not true, then there must be one or more counterexamples—positive integers n that make $S(n)$ false. Let n_0 be the smallest counterexample. Certainly $n_0 \neq 1$, since the proof of the base case explicitly showed that $S(1)$ is true. On the other hand, if $n_0 > 1$, then look at the inductive step, with $k = n_0 - 1$; it was proved that $S(n_0 - 1) \rightarrow S(n_0)$. Since n_0 is the *smallest* counterexample, the inductive hypothesis $S(n_0 - 1)$ is true. Therefore the conclusion of the inductive step must also be true, namely $S(n_0)$. But this contradicts our assumption that n_0 was a counterexample to $\forall n{:}\, S(n)$. Therefore $\forall n{:}\, S(n)$ has no counterexamples, and our argument is complete.

29. In each case ($i = 1, 2, 3$) we find the correct formula by looking at the pattern that appears when we calculate $A(n, i)$ iteratively for successive values of n. Because of the recursive definition, this computation uses the results for $i - 1$.

(a) The sequence begins 2, 3, 4, 5, ..., so we conjecture that $A(n, 1) = n + 2$. For the base case $A(0, 1) = A(1, 0) = 1 + 1 = 2 = 0 + 2$. Assume the inductive hypothesis, that $A(k, 1) = k + 2$. Then

$$\begin{aligned} A(k+1, 1) &= A(A(k, 1), 0) && \text{(by definition)} \\ &= A(k+2, 0) && \text{(by the inductive hypothesis)} \\ &= k + 2 + 1 && \text{(by definition)} \\ &= (k+1) + 2. \end{aligned}$$

(b) The sequence begins 3, 5, 7, 9, ..., so we conjecture that $A(n, 2) = 2n + 3$. For the base case $A(0, 2) = A(1, 1) = 1 + 2 = 3$ by part (a). Assume the inductive hypothesis, that $A(k, 2) = 2k + 3$. Then

$$\begin{aligned} A(k+1, 2) &= A(A(k, 2), 1) && \text{(by definition)} \\ &= A(2k+3, 1) && \text{(by the inductive hypothesis)} \\ &= (2k+3) + 2 && \text{(by part (a))} \\ &= 2(k+1) + 3. \end{aligned}$$

(c) The sequence begins 5, 13, 29, 61, ..., so we conjecture that $A(n,3) = 2^{n+3} - 3$. For the base case $A(0,3) = A(1,2) = 2 \cdot 1 + 3 = 5$ by part (b), and $2^{0+3} - 3 = 5$. Assume the inductive hypothesis, that $A(k,3) = 2^{k+3} - 3$. Then

$$\begin{aligned} A(k+1,3) &= A(A(k,3),2) && \text{(by definition)} \\ &= A(2^{k+3} - 3, 2) && \text{(by the inductive hypothesis)} \\ &= 2(2^{k+3} - 3_+3 && \text{(by part (b))} \\ &= 2^{(k+1)+3} - 3. \end{aligned}$$

31. We claim that f_n is odd when $n \equiv 0$ or $1 \pmod 3$, and f_n is even when $n \equiv 2 \pmod 3$. In other words, f_n is even if and only if $n \equiv 2 \pmod 3$. The base cases are easily checked: f_0 and f_1 are odd. Assume the inductive hypothesis, that for all $i < k$, f_i is even if and only if $i \equiv 2 \pmod 3$. We must show that f_k is even if and only if $k \equiv 2 \pmod 3$. Now if $k \equiv 2 \pmod 3$, then $k - 1 \equiv 1 \pmod 3$ and $k - 2 \equiv 0 \pmod 3$, so by the inductive hypothesis, f_{k-1} and f_{k-2} are both odd. Hence $f_k = f_{k-1} + f_{k-2}$ is even. If $k \equiv 0 \pmod 3$, then $k - 1 \equiv 2 \pmod 3$ and $k - 2 \equiv 1 \pmod 3$, so by the inductive hypothesis, f_{k-1} is even and f_{k-2} is odd. Hence $f_k = f_{k-1} + f_{k-2}$ is odd. The case in which $k \equiv 1 \pmod 3$ is similar to this last case.

33. **(a)** This can most easily be proved by simple algebra (factoring), without mathematical induction. Indeed, $f_{n+2}^2 - f_{n+1}^2 = (f_{n+2} - f_{n+1})(f_{n+2} + f_{n+1}) = f_n f_{n+3}$ by the definition of the Fibonacci sequence.

(b) We want to prove that $f_{n+1}^2 - f_n f_{n+2} = (-1)^{n+1}$ for all n. The base case ($n = 0$) is clear: $f_1^2 - f_0 f_2 = 1 - 2 = (-1)^1$. Assume the inductive hypothesis, that $f_{k+1}^2 - f_k f_{k+2} = (-1)^{k+1}$. Then

$$\begin{aligned} f_{k+2}^2 - f_{k+1} f_{k+3} &= f_{k+2}^2 - f_{k+1}(f_{k+2} + f_{k+1}) && \text{(by definition)} \\ &= f_{k+2}^2 - f_{k+1} f_{k+2} - f_{k+1}^2 \\ &= f_{k+2}^2 - f_{k+1} f_{k+2} - \left(f_k f_{k+2} + (-1)^{k+1}\right) \\ & && \text{(by the inductive hypothesis)} \\ &= f_{k+2}^2 - f_{k+2}(f_{k+1} + f_k) - (-1)^{k+1} && \text{(by factoring)} \\ &= f_{k+2}^2 - f_{k+2} f_{k+2} - (-1)^{k+1} && \text{(by definition again)} \\ &= -(-1)^{k+1} = (-1)^{k+2}. \end{aligned}$$

35. **(a)** Let $n \in \mathbf{N}$ be the number of numbers being added, which are denoted a_1, a_2, ..., a_n. For the base case, $n = 0$, the empty sum is 0 by definition, and 0 is even. Assume the inductive hypothesis, that the sum of k even numbers is even. Then the sum $(a_1 + a_2 + \cdots + a_k) + a_{k+1}$ is the sum of two even numbers (by the inductive hypothesis), which is even by Exercise 2 in Section 1.3.

(b) Let $2n$ be the number of odd numbers being added, which are denoted a_1, a_2, $\ldots$, a_{2n}. For the base case, $n = 0$, the empty sum is 0 by definition, and 0 is even. Assume the inductive hypothesis, that the sum of $2k$ odd numbers is even. Then the sum $(a_1 + a_2 + \cdots + a_{2k}) + a_{2k+1} + a_{2k+2}$ is the sum of an even number and two odd numbers (by the inductive hypothesis), which is the sum of an even number and an even number, which is even (by two applications of Exercise 2 in Section 1.3).

(c) Let $2n + 1$ be the number of odd numbers being added, which are denoted a_1, a_2, $\ldots$, a_{2n+1}. For the base case, $n = 0$, the sum of one odd number a_1 is itself, which is odd. Assume the inductive hypothesis, that the sum of $2k+1$ odd numbers is odd. Then the sum $(a_1 + a_2 + \cdots + a_{2k+1}) + a_{2k+2} + a_{2k+3}$ is the sum of an odd number and two odd numbers (by the inductive hypothesis), which is the sum of an odd number and an even number, which is odd (by two applications of Exercise 2 in Section 1.3).

37. (a) For the base case ($n = 0$) we have $3 \mid 6$. Assume the inductive hypothesis, that $3 \mid k^3 - 4k + 6$. Then

$$\begin{aligned}(k+1)^3 - 4(k+1) + 6 &= k^3 + 3k^2 + 3k + 1 - 4k - 4 + 6 \\ &= (k^3 - 4k + 6) + (3k^2 + 3k - 3)\,.\end{aligned}$$

The first summand in parentheses in the final expression is divisible by 3 by the inductive hypothesis, and the second is clearly divisible by 3, so $3 \mid (k+1)^3 - 4(k+1) + 6$, as desired.

(b) We write $n^3-4n+6 = n^3-n-3n+6 = n(n^2-1)-3(n-2) = (n-1)n(n+1)-3(n-2)$. Now the first term in the final expression is divisible by 3 since one of the three consecutive numbers $n-1$, n, and $n+1$ is necessarily a multiple of 3. Clearly the second term is divisible by 3. The result follows.

39. (a) We prove by induction on m that if the game table consists of n stones and $n - m$ piles, then exactly m plays remain. The base case is $m = 0$. If there are $n - 0 = n$ piles, then clearly each pile contains one stone, so no plays remain. Assume the inductive hypothesis, and suppose that there are $n - (m+1) = n - m - 1$ piles. Whatever a player P now does, the number of piles must increase by 1, leaving $n - m$ piles. By the inductive hypothesis, m plays remain after that. Thus counting P's move, $m + 1$ plays remained when there were $n - (m+1)$ piles left. Thus the statement is true for all m. Letting $m = n - 1$, we obtain the desired conclusion, that $n - 1$ plays remain when there is one pile.

(b) The model described here gives the same game. Since there are $n-1$ spaces between adjacent stones, the game lasts exactly $n - 1$ moves.

(c) If n is odd, then the second player wins, since an even number $(n - 1)$ of moves remains. If n is even, then the first player wins.

41. The base case is $v = \lambda$. We have $(uv)^R = (u\lambda)^R = u^R = \lambda u^R = \lambda^R u^R$. Assume the inductive hypothesis, that the statement is true for second strings of length k, and let $v = \alpha x$, where α is a string of length k and x is a symbol. Then

$$\begin{aligned}(uv)^R = (u\alpha x)^R &= ((u\alpha)x)^R \\ &= x(u\alpha)^R \qquad \text{(by the definition of } R\text{)} \\ &= x(\alpha^R u^R) \qquad \text{(by the inductive hypothesis)} \\ &= (x(\alpha^R))u^R \\ &= (\alpha x)^R u^R \qquad \text{(by the definition of } R\text{)} \\ &= v^R u^R .\end{aligned}$$

43. Let $S(n)$ be the statement that any solution to the problem of transferring n disks from one given peg to another (different) given peg, according to the rules of the towers of Hanoi, requires at least $2^n - 1$ moves. We need to prove $\forall n \geq 1 : S(n)$. The base case is clear, since if $n = 1$, then $2^1 - 1 = 1$ move is required. Assume the inductive hypothesis, that any solution to the problem of transferring k disks from one peg to another requires at least $2^k - 1$ moves. Now in order to solve the problem with $k + 1$ disks, we must at some point move disk $k + 1$. This can happen only when all the other disks have been moved off the peg on which this largest disk sits, onto one of the other pegs. By the inductive hypothesis, this requires at least $2^k - 1$ moves. Similarly, after disk $k + 1$ has been transferred to the desired peg (for the last time, if it moved more than once), the other k disks must make their way back onto the peg on which disk $k + 1$ now sits, and again by the inductive hypothesis, this requires at least $2^k - 1$ moves. Thus the total number of moves required is at least $(2^k - 1) + 1 + (2^k - 1) = 2 \cdot 2^k - 1 = 2^{k+1} - 1$, as desired.