and multiply by 100) yields $x \approx 996$. (There is also a root near $x = 1.007$, which does not concern us.) Therefore $n = 2^x \approx 2^{996} \approx 10^{300}$. Thus $n^{0.01}$ overtakes $\log n$ around 10^{300}.

27. Let $f(n) = 2^{\sqrt{n}}$. Then to evaluate the limit of $f(n)/k^n$, take the logarithm and note that $\sqrt{n} - n\log k \to -\infty$ as $n \to \infty$. Thus the limit is 0, so $f \in O(k^n)$. On the other hand, a similar evaluation of the limit of $f(n)/n^k$ (since $\sqrt{n} - k\log n \to \infty$) shows that $f \notin O(n^k)$.

SECTION 4.4 Intractable and Unsolvable Problems

1. (a) Here is a short proof that the answer is yes (this is a nontautology): $\overline{\mathrm{T}} \vee (\mathrm{F} \wedge \mathrm{T})$ is F.

   (b) Here is a long proof that the answer is no (this is in fact a tautology). If either P or Q is F, then the given proposition is T, because of the disjunction with $\overline{P}$ and $\overline{Q}$. Otherwise, P and Q are both true, so $P \wedge Q$ is T, whence $(P \wedge Q) \vee R$ is T, so again the proposition is true. (We could also have drawn the truth table.)

3. In each case we let n be the number of digits in N.

   (a) Given $k = N/2$ as a hint, we can easily verify that $N = 2k$ by a multiplication of k by 2. Since k has about n digits, this takes only $O(n)$ steps (using the usual grade-school multiplication algorithm). Even more quickly, we can ignore any hint and just check the last digit of N.

   (b) Given $k = N/3$ as a hint, we can easily verify that $N = 3k$ by a multiplication of k by 3. Since k has about n digits, this again takes only $O(n)$ steps.

   (c) Given $k = \sqrt{N}$ as a hint, we can easily verify that $N = k^2$ by a multiplication of k by k. Since k has about $n/2$ digits, this only takes $O(n^2)$ steps (using the usual grade-school multiplication algorithm—see Section 4.5).

   (d) Given i, j, k, and l as a hint, we can easily verify that $N = i^2 + j^2 + k^2 + l^2$ in $O(n^2)$ steps, since each of the four numbers has no more than n digits.

5. We calculate by hand or on a computer with enough digits of accuracy that $193707721 \times 761838257287 = 147573952589676412927 = 2^{67} - 1$. Thus $2^{67} - 1$ has nontrivial factors and so is not prime.

7. (a) Given a hint as to which sets form the pairwise disjoint collection, we form their pairwise intersections (there are only $k(k-1)/2$ of them) to verify this.

   (b) Given a hint as to the partition, we sum each half to verify that the sums are equal.

   (c) Given such a C as a hint, which we can assume has no more than $2S$ elements, we can take each pair in S and verify the condition by searching C.

9. Generate each subset of S (there are $2^{|S|}$ such subsets), and check whether its sum is g. Since this takes at least $2^{|S|}$ steps, it is a bad algorithm.

11. Sort $S = \{s_1, s_2, \ldots, s_n\}$ so that $s_1 < s_2 < \cdots < s_n$. If x is the sum of a subset of S, then we can find this subset by the following algorithm.

```
for i ← n down to 1 do
    if s_i < x then
        begin
            x ← x − s_i
            put s_i into the subset
        end
```

At this point $x = 0$ if and only if the original x was a sum of some subset of S. The reason that this procedure works is that $s_i > \sum_{j=1}^{i-1} s_j$, so that if $s_i < x$ when we are looking at s_i, then we know that we need to include s_i in the subset. Clearly this algorithm has $O(n)$ complexity.

13. Construct the following algorithm for the one-variable case. The input is the formula for f, where $f(x) = 0$ is the given equation. Set n equal to 0. Repeatedly calculate $f(n)$ and $f(-n)$ and then add 1 to n. If you ever find that $f(n) = 0$ or $f(-n) = 0$, then halt. Now we encode this algorithm and its input (f) and give it to a purported algorithm for the halting problem. It answers yes if and only if the equation $f(x) = 0$ has an integral solution. Thus the solution to the halting problem gives a solution to the polynomial problem. The multivariable case is only a little harder—the idea is the same. In this case, the loop must test systematically for all solutions of $f(x_1, x_2, \ldots, x_k)$. For each n, there are only a finite number of k-tuples in which the absolute value of each coordinate is at most n. For the nth pass through the loop, then, the algorithm checks all of these k-tuples, again halting if and when it finds a solution.

SECTION 4.5 Algorithms for Arithmetic and Algebra

1. **(a)** $carry = 0$; $c_0 = 5 + 9 + 0 = 14$, $c_0 = 4$, $carry = 1$; $c_1 = 7 + 5 + 1 = 13$, $c_1 = 3$, $carry = 1$; $c_2 = 3 + 2 + 1 = 6$, $c_2 = 6$, $carry = 0$; $c_3 = 0$; answer is 0634

(b) $carry = 0$; $c_0 = 9 + 9 + 0 = 18$, $c_0 = 8$, $carry = 1$; $c_1 = 9 + 9 + 1 = 19$, $c_1 = 9$, $carry = 1$; $c_2 = 9 + 0 + 1 = 10$, $c_2 = 0$, $carry = 1$; $c_3 = 1$; answer is 1089

3. We use a calculator and the rules of exponents to break these calculations down to manageable size.

(a) $2^{99} = (2^{24})^4 \cdot 2^3 = 16777216^4 \cdot 2^3 \equiv 16^4 \cdot 8 = 65536 \cdot 8 \equiv 36 \cdot 8 = 288 \equiv 88 \pmod{100}$

(b) $98^{10} \equiv (-2)^{10} = 1024 \equiv 24 \pmod{100}$

(c) $52^{52} = (52^2)^{26} = 2704^{26} \equiv 4^{26} = (4^{13})^2 = (2^{13})^4 = 8192^4 \equiv (-8)^4 = 4096 \equiv 96$ (mod 100)

5. (a) For the sum, $c_0 = -2+5 = 3$, $c_1 = 4+1 = 5$, $c_2 = 3+0 = 3$, so the sum is $3+5x+3x^2$. For the product, we have $c_0 = 0+(-2)\cdot 5 = -10$, $c_1 = 0+(-2)\cdot 1 = -2$, $c_1 = -2+4\cdot 5 = 18$, $c_2 = 0+4\cdot 1 = 4$, $c_2 = 4+3\cdot 5 = 19$, $c_3 = 0+3\cdot 1 = 3$, so the product is $-10+18x+19x^2+3x^3$.

   (b) For the sum, $c_0 = 0+0 = 0$, $c_1 = 0+0 = 0$, $c_2 = 3+(-3) = 0$, so the sum is $0+0x+0x^2$, i.e., the zero polynomial. For the product, we have $c_0 = 0+0\cdot 0 = 0$, $c_1 = 0+0\cdot 0 = 0$, $c_2 = 0+0\cdot(-3) = 0$, $c_1 = 0+0\cdot 0 = 0$, $c_2 = 0+0\cdot 0 = 0$, $c_3 = 0+0\cdot(-3) = 0$, $c_2 = 0+3\cdot 0 = 0$, $c_3 = 0+3\cdot 0 = 0$, $c_4 = 0+3\cdot(-3) = -9$, so the product is $0+0x+0x^2+0x^3-9x^4 = -9x^4$.

7. We add entry by entry, obtaining

$$A+B = \begin{bmatrix} 3+5 & 8+(-8) & -2+(-4) & 3+6 \\ 0+4 & 0+0 & 2+(-2) & -1+11 \end{bmatrix} = \begin{bmatrix} 8 & 0 & -6 & 9 \\ 4 & 0 & 0 & 10 \end{bmatrix}.$$

9. (a) $24_{\text{six}} = 4+2\cdot 6 = 16$ (b) $301_{\text{six}} = 1+3\cdot 6^2 = 109$

   (c) $2_{\text{six}} = 2$ (d) $10000_{\text{six}} = 1\cdot 6^4 = 1296$

11. We apply Horner's method to evaluate $d_0 + d_1\beta + d_2\beta^2 + \cdots + d_{n-1}\beta^{n-1}$.

```
procedure to_base_ten(β : integer ≥ 2, d_{n-1}d_{n-2}...d_1d_0 : string of digits base β)
  {computes the value of the base β numeral d_{n-1}d_{n-2}...d_1d_0 }
    x ← d_{n-1}
    for i ← n − 2 down to 0 do
        x ← β · x + d_i
    return(x)
```

13. We subtract digit by digit, from right to left, "borrowing" from the next column as needed.

procedure *difference*(x, y : natural numbers)
{ x and y are represented by the decimal numerals $a_{n-1} \ldots a_0$ and $b_{m-1} \ldots b_0$, respectively; the answer $z = x - y$ will appear as the decimal numeral $c_{k-1} \ldots c_0$, where $k = \max(n, m)$; we assume that a_i is defined to be 0 for $i \geq n$, that b_i is defined to be 0 for $i \geq m$, and that $x \geq y$ }

```
k ← max(n, m)
borrow ← 0
for i ← 0 to k − 1 do
    begin
        c_i ← a_i − b_i − borrow
        if c_i < 0 then {need to borrow 10 from next column}
                begin
                    c_i ← c_i + 10
                    borrow ← 1
                end
            else borrow ← 0
    end
return(z)
```

The time complexity of the algorithm is $O(k) = O(n+m)$, since the main loop is iterated k times.

15. $2^{340} = (2^{10})^{34} = 1024^{34} \equiv 1^{34} = 1 \pmod{341}$

17. If we take a number between 0 and 1, multiply it by N, and round down to an integer, we will have an integer between 0 and $N-1$, inclusive. We add 1 to get an integer in the desired range.

procedure *random_integer*(N : positive integer)
{computes a random integer between 1 and N, inclusive; assume that *rand* returns a random real number between 0 and 1 each time it is called}

```
x ← rand
return(1 + ⌊x · N⌋)
```

19. (a) **procedure** *matrix_add*(A, B)
{ $A = (a_{ij})$ and $B = (b_{ij})$ are m by n matrices; the sum is the m by n matrix $C = (c_{ij})$ }

```
for i ← 1 to m do
    for j ← 1 to n do
        c_ij ← a_ij + b_ij {add entry by entry}
return(C)
```

(b) **procedure** *matrix_multiply*(A, B)

```
{ A = (a_ik) is an m by n matrix and B = (b_kj) is an n by p matrix;
  the product is the m by p matrix C = (c_ij) }
  for i ← 1 to m do
      for j ← 1 to p do
          begin {compute (i,j)th entry as a sum of products}
              c_ij ← 0
              for k ← 1 to n do
                  c_ij ← c_ij + a_ik b_kj
          end
  return(C)
```

21. The matrix all of whose entries are 0 has this property, since $a_{ij} + 0 = a_{ij}$.

23. Since $2^{24} = 16777216 \equiv 16 \not\equiv 1 \pmod{25}$, we know that 25 is not prime.

25. It is faster to multiply $A \times B$ first when $pqr + prs < qrs + pqs$ (see Exercise 20). We divide through by $pqrs$ to obtain the nice condition $s^{-1} + q^{-1} < p^{-1} + r^{-1}$.

27. (a) **procedure** *slow*$(a_1, a_2, \ldots, a_n$: sequence of real numbers)

```
{naively finds the sum of all products a_i a_j for i < j }
  sum ← 0
  for i ← 1 to n − 1 do
      for j ← i + 1 to n do
          sum ← sum + a_i a_j
  return(sum)
```

The complexity of this algorithm is clearly proportional to n^2.

(b) If we expand $(a_1 + a_2 + \cdots + a_n)^2$, we obtain $a_1^2 + a_2^2 + \cdots + a_n^2 + 2 \cdot$ (desired sum). We can therefore compute the answer as $((a_1 + a_2 + \cdots + a_n)^2 - a_1^2 - a_2^2 - \cdots - a_n^2)/2$, and this takes only $O(n)$ steps.

CHAPTER 5
INDUCTION AND RECURSION

SECTION 5.1 Recursive Definitions

1. In Example 1 we found that $f_7 = 21$ and $f_8 = 34$. Hence $f_9 = 34 + 21 = 55$, $f_{10} = 55 + 34 = 89$, and $f_{11} = 89 + 55 = 144$.

3. Each term is the sum of the three preceding terms. Thus the sequence begins 1, 1, 1, 3, 5, 9, 17, 31, 57, 105. In particular, $a_{10} = 105$.

5. (a) $A^2 = A \times A = \begin{bmatrix} -2 & -3 \\ 9 & 1 \end{bmatrix}$

(b) $A^0 = I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

(c) $A^3 = A \times A^2 = \begin{bmatrix} -11 & -4 \\ 12 & -7 \end{bmatrix}$

(d) $A^1 = A = \begin{bmatrix} 1 & -1 \\ 3 & 2 \end{bmatrix}$

7. (a) yes (unsigned integer)

(b) no (no operator)

(c) no (the raised dot is not a legal symbol)

(d) no (no parentheses)

(e) yes ($x2$ is a variable name)

(f) no (needs more parentheses)

9. On day 0 we get, among other things, 1, s, and 3. On day 1, (-1), $(-s)$, and $(3+1)$ come into being. On day 2 we see the birth of $((-1)-s)$ and $((-s)*(3+1))$. Finally, on day 3 we can construct $(((-1)-s)+((-s)*(3+1)))$.

11. **(a)** $P(1) = 1$, $P(2) = P(2 - P(1)) + 1 = P(2-1) + 1 = P(1) + 1 = 1 + 1 = 2$, $P(3) = P(3-2) + 1 = 2$, $P(4) = P(4-2) + 1 = 3$, $P(5) = P(5-3) + 1 = 3$, $P(6) = 3$, $P(7) = 4$, $P(8) = 4$, $P(9) = 4$, $P(10) = 4$, $P(11) = 5$, $P(12) = 5$, $P(13) = 5$, $P(14) = 5$, $P(15) = 5$, $P(16) = 6$

(b) For this definition to be valid, $n - P(n-1)$ must be a positive integer less than n for all $n \geq 2$, so that $P(n - P(n-1))$ makes sense and has already been defined. There is no obvious reason that this will be true.

(c) one 1, two 2's, three 3's, four 4's, etc.

13. **(a)** $x^0 = 1$, $x^{n+1} = x \cdot x^n$

(b) Let $I(x, n) = x^{x^{\cdot^{\cdot^{x}}}}$, with n x's in the exponent ($n+1$ x's in all). The recursive definition is that $I(x,0) = x$, and $I(x, n+1) = x^{I(x,n)}$. Letting $x = 0$, we see that $I(0,0) = 0$, $I(0,1) = 0^0 = 1$, $I(0,2) = 0^1 = 0$, $I(0,3) = 0^0 = 1$, and so on. Thus $I(0,n)$ is 0 when n is even and 1 when n is odd.

15. **(a)** We apply the recursive definition repeatedly, finally using the base case $\varphi(x,y,0) = x + y$ three times:

$$\begin{aligned}
\varphi(3,3,1) &= \varphi(3, \varphi(3,2,1), 0) = \varphi(3, \varphi(3, \varphi(3,1,1), 0), 0) \\
&= \varphi(3, \varphi(3, \varphi(3, \varphi(3,0,1), 0), 0), 0) \\
&= \varphi(3, \varphi(3, \varphi(3,0,0), 0), 0) = \varphi(3, \varphi(3,3,0), 0) \\
&= \varphi(3,6,0) = 9
\end{aligned}$$

(b) For the last step here, we use the result from part (a).

$$\begin{aligned}
\varphi(3,2,2) &= \varphi(3, \varphi(3,1,2), 1) = \varphi(3, \varphi(3, \varphi(3,0,2), 1), 1) \\
&= \varphi(3, \varphi(3,1,1), 1) = \varphi(3, \varphi(3, \varphi(3,0,1), 0), 1) \\
&= \varphi(3, \varphi(3,0,0), 1) = \varphi(3,3,1) = 9
\end{aligned}$$

17. We apply the recursive definition repeatedly, using the base case $A(x,0) = x + 1$ six times:

$$\begin{aligned}
A(1,2) &= A(A(0,2), 1) = A(A(1,1), 1) \\
&= A(A(A(0,1), 0), 1) = A(A(A(1,0), 0), 1) \\
&= A(A(2,0), 1) = A(3,1) \\
&= A(A(2,1), 0) = A(A(A(1,1), 0), 0) \\
&= A(A(A(A(0,1), 0), 0), 0) = A(A(A(A(1,0), 0), 0), 0) \\
&= A(A(A(2,0), 0), 0) = A(A(3,0), 0) = A(4,0) = 5
\end{aligned}$$

(b) For the base case (first line), $A(n,0)$ is defined directly. For all other cases (second and third lines) $A(n,i)$ is defined in terms of values of $A(n',i')$, where (n',i') precedes (n,i) in right to left lexicographical order, i.e., either $i' < i$ or else $i' = i$ and $n' < n$.

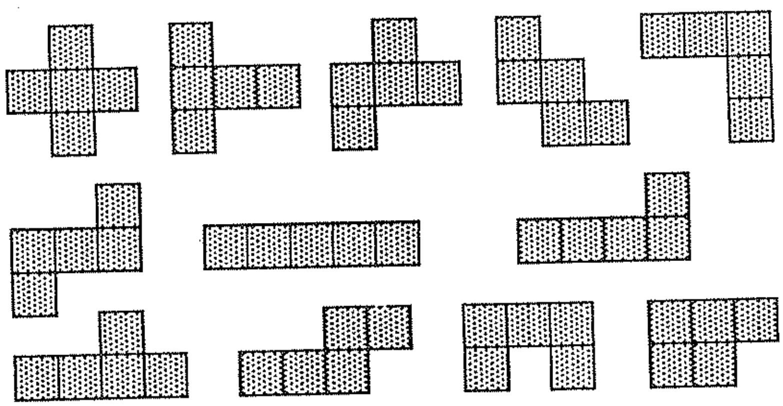
19. Variable names, 0, and 1 are Boolean expressions. If α and β are Boolean expressions, then so are (α), $\overline{\alpha}$, $\alpha\beta$, $\alpha \cdot \beta$, and $\alpha + \beta$. Note that parentheses are allowed but not required.

21. (a) $a_1 = 1$, $a_2 = 0.5$, $a_3 = 0.6666667$, $a_4 = 0.6$, $a_5 = 0.625$, $a_6 = 0.6153846$, $a_7 = 0.6190476$, $a_8 = 0.6176471$, $a_9 = 0.6181818$, $a_{10} = 0.6179775$, $a_{11} = 0.6180556$, $a_{12} = 0.6180258$, $a_{13} = 0.6180371$, $a_{14} = 0.6180328$, $a_{15} = 0.6180344$, $a_{16} = 0.6180338$

(b) The sequence $\{a_n\}$ seems to be converging to a specific value around 0.618. In fact, the value is $(-1+\sqrt{5})/2$, which is the positive root of the equation $x = (1+x)^{-1}$.

23. (a) There is clearly only one 4-nomino with four squares in a row. If a 4-nomino has three squares in a row, then the other square can only be attached in two locations (up to rotation and flipping), and the result of each of these possibilities is shown. Otherwise, the 4-nomino must start with three squares in an L shape, and there are only two different places to add the fourth square without creating four in a row; both of these pictures are shown.

(b) There are 12 5-nominoes, as shown here.



25. (a) We exhibit the new lists and new insides that arise on each day.

	new lists	new insides
day 0	( )	5
day 1	(5)	( ); 5, 5
day 2	(( )); (5, 5)	(5); ( ), ( ); ( ), 5, 5; 5, 5, ( ); 5, 5, 5, 5; ( ), 5; 5, ( ); 5, 5, 5

(b) We exhibit the relevant new lists and new insides that arise on each day.

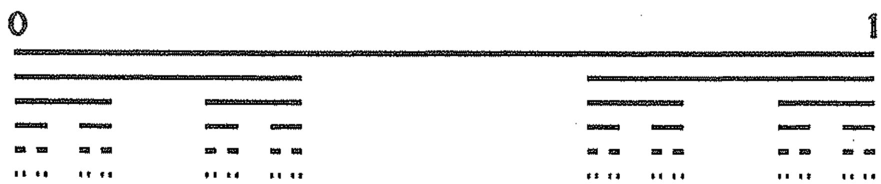
	new lists	new insides
day 0	( )	3; 4; 7; 2
day 1	(4)	( ); 7,2
day 2	(7,2)	( ),( ); (4)
day 3	(( ),( ))	(7,2); 3,(4)
day 4		(( ),( )); 3,(4),(7,2)
day 5		3,(4),(7,2),(( ),( ))
day 6	(3,(4),(7,2),(( ),( )))	

27. **(a)** day 0: 5, 7; day 1: $5-1=4$, $7-1=6$, $5+5=10$, $5+7=12$, $7+7=14$; day 2: $4-1=3$, $4+4=8$, $4+5=9$ (also $10-1=9$), $4+7=11$, $14-1=13$, $10+5=15$, $10+6=16$, $10+7=17$, $12+6=18$, $12+7=19$, $10+10=20$, $14+7=21$, $10+12=22$, $12+12=24$, $12+14=26$, $14+14=28$

(b) Since $x \in E \to 2x \in E$, and since $5 \in E$, we see that E is unbounded (has arbitrarily large elements). Since $x \in E \to x-1 \in E$, we see that E has no gaps below any number. Therefore $E = \mathbf{Z}$.

29. If d is a digit, then $d.$ and $.d$ are unsigned decimals. If d is a digit and α is an unsigned decimal, then $d\alpha$ and αd are unsigned decimals. If α is an unsigned decimal, then $+\alpha$ and $-\alpha$ are signed decimals.

31. **(a)** For good measure we have shown C_0, C_1, C_2, C_3, C_4, and C_5.



(b) One segment of length $1/3$ is removed; two segments of length $1/9$ are removed; four segments of length $1/27$ are removed, and so on. Therefore the total length of the removed segments is

$$\frac{1}{3}+\frac{2}{9}+\frac{4}{27}+\frac{8}{81}+\cdots=\frac{1}{3}\left(1+\frac{2}{3}+\left(\frac{2}{3}\right)^2+\left(\frac{2}{3}\right)^3+\cdots\right)=\frac{1}{3}\cdot\frac{1}{1-(2/3)}=1.$$

(c) The Cantor set contains the numbers $1/3$, $1/9$, $1/27$, $\ldots$, since these are never removed.

(d) Use base 3 notation. The removed segments are those values whose "decimals" (base 3) must contain a 1. (For example, the numbers between $1/3$ and $2/3$ have a 1 in the first digit after the "decimal" point in their base 3 expansions.) Thus all those base 3 "decimals" containing only 0's and 2's remain. There are just as many of these as there are base 2 "decimals" with 0's and 1's (replace each 2 by a 1), which represent all the numbers in the interval $[0,1]$ (and then some, since the representations are not unique). Since this latter set is uncountable, we are done.

33. **(a)** A game is winning for Righty if and only if there exists a game in R that is either winning for Righty or winning for the second player and every game in L is either winning for Righty or winning for the first player. A game is winning for the first player if and only if there exists a game in L that is either winning for Lefty or winning for the second player and there exists a game in R that is either winning for Righty or winning for the second player. A game is winning for the second player if and only if every game in L is either winning for Righty or winning for the first player and every game in R is either winning for Lefty or winning for the first player. (The fourth case was given in the hint.)

(b) The game 0 is winning for the second player. The games $*$ and $(\{*,1\},\{-1\})$ are winning for the first player. The games 1, $(\{0\},\{1\})$, and $(\{1\},\emptyset)$ are winning for Lefty. The game -1 is winning for Righty.

SECTION 5.2 Recursive Algorithms

1. We indent the moves to show the levels of recursion.

        Move disk 1 from peg A to peg B.
    Move disk 2 from peg A to peg C.
        Move disk 1 from peg B to peg C.
Move disk 3 from peg A to peg B.
        Move disk 1 from peg C to peg A.
    Move disk 2 from peg C to peg B.
        Move disk 1 from peg A to peg B.

3. We need to concatenate the last symbol with the reverse of the first $n-1$ symbols.

procedure *reverse*(s : string)
{assume that $s = s_1s_2\ldots s_n$, with $n \ge 0$}
if $n \le 1$ **then return**(s)
else return($s_n\mathit{reverse}(s_1s_2\ldots s_{n-1})$)

5. Assume that there is a new element ∞ at the end of each list. We start by looking at the first elements of the two lists. Since $1 \in B$ is less than $3 \in A$, we make 1 the first element in the merged list and then move to consider $6 \in B$. At this point we are comparing $3 \in A$ with $6 \in B$. Since 3 is smaller, we put it as the second element in the merged list and then consider $5 \in A$ versus $6 \in B$. We continue in this way until both original lists are exhausted (in this case, after 11 steps). Ties can be broken arbitrarily, by putting either of the elements into the new list (although Algorithm 8 actually chooses the element from B in such a case). The final merged list is $(1,3,5,6,6,9,9,20,24,31,49)$.