rod of length n. For the sample price chart appearing in Figure 14.1, the call EXTENDED-BOTTOM-UP-CUT-ROD$(p, 10)$ returns the following arrays:

i	0	1	2	3	4	5	6	7	8	9	10
$r[i]$	0	1	5	8	10	13	17	18	22	25	30
$s[i]$		1	2	3	2	2	6	1	2	3	10

A call to PRINT-CUT-ROD-SOLUTION$(p, 10)$ prints just 10, but a call with $n = 7$ prints the cuts 1 and 6, which correspond to the first optimal decomposition for r_7 given earlier.

```
EXTENDED-BOTTOM-UP-CUT-ROD(p, n)
let r[0 : n] and s[1 : n] be new arrays
r[0] = 0
for j = 1 to n                    // for increasing rod length j
    q = −∞
    for i = 1 to j                // i is the position of the first cut
        if q < p[i] + r[j − i]
            q = p[i] + r[j − i]
            s[j] = i              // best cut location so far for length j
    r[j] = q                      // remember the solution value for length j
return r and s
```

```
PRINT-CUT-ROD-SOLUTION(p, n)
(r, s) = EXTENDED-BOTTOM-UP-CUT-ROD(p, n)
while n > 0
    print s[n]          // cut location for length n
    n = n − s[n]        // length of the remainder of the rod
```

Exercises

14.1-1

Show that equation (14.4) follows from equation (14.3) and the initial condition $T(0) = 1$.

14.1-2

Show, by means of a counterexample, that the following "greedy" strategy does not always determine an optimal way to cut rods. Define the ***density*** of a rod of length i to be p_i / i, that is, its value per inch. The greedy strategy for a rod of length n cuts off a first piece of length i, where $1 \le i \le n$, having maximum