Problem 2.10

The idea in dynamic programming is often to obtain an accumulative return function, say $f_n(\mathbf{S}_n)$, where the accumulative return at stage n is the return at stage n plus the accumulative return at stage $n-1$. If we know the best accumulative return at stage $n-1$, it follows that

```
f(n, S)  = max{Log(c) + b * f(n - 1, S - c)}
```

where b is the discount factor used to reduce the value of the future accumulated returns. $S - c$ is the remaining capital to spend for periods $n-1, n-2, \ldots, 1$. The essential recursive function to write is:

```
Function f(n, S)
fStar = -1
If n = 1 Then
    f = Log(S)
    cStar = S
Else
    For c = 1 To S - n + 1
        fTest = Log(c) + b * f(n - 1, S - c)
        If fTest > fStar Then
            fStar = fTest
            cStar = c
        End If
    Next c
    f = fStar
End If
Exit Function
```

Recall that for the last period (stage $n = 1$) we consume whatever is left. We have to spend minimum 1 unit per period. The function above did not perform any bookkeeping. The optimal consumption in stage $N = 5$, i.e., the first period, is found from Table 1 and shows that 3 units should be

Table 1: Calculations for stage $N = 5$

S_4	$f_4^*(S_4)$	c_5	$u(c_5) = \ln(c_5)$	$f_5(10, c_5) = \ln(c_5) + bf_4^*(S_4)$
4	0	6	1.79	1.79
5	0.69	5	1.61	2.16
6	1.25	4	1.39	2.39
7	1.69	3	1.1	2.45
8	2.1	2	0.69	2.37
9	2.45	1	0	1.96

consumed. Running the code shows that it is optimal to consume 2 units for the 3 following periods, and one unit for the last period.