Problem 1 (Problem 15.1-2)

Solution: Here is a counterexample to prove that greedy algorithm does not provide an optimal solution every time, Let p_1=1, p_2=20, p_3=33, p_4=36. Let the length of the rod be 4 inches. When we run the greedy algorithm, the first cut for the rod would be of length 3 whose density is maximum. As a results, the total price would be 34. However, if we cut the rod into two pieces of length 2, the total price would be 40 and this is the optimal solution. Therefore, the greedy algorithm fails in this counterexample.

Problem 2 (Problem 15.1-3)

Solution:

1) We use dynamic programming to solve this problem. Let r_n be the max revenue for length n. Then we have $r_n = max_{1 \le i < n}\{p_n, p_i + r_{n-i} - c\}$

2) Pseudocode:

```
Input: p, n, c
Output: $r_n$
Let $r_0$=0
For j from 1 to n do
    Let q = $p_j$
    For i from 1 to j-1 do
        Let $q = max\{q, p_i + r_{j-i} - c\}$
    End for
    Let $r_j = q$
End for
```

3) Prove:

In this program, $r_{1,2\dots,n-1,n}$ store the maximum revenue of cutting rod with length ranging from 1 to n. The core of this algorithm is the recursive part $r_n = max_{1 \le i < n}\{p_n, p_i + r_{n-i} - c\}$. We loop from length 1 to n, for each length, we calculate the maximum revenue based on the previous solution. When the rod with length of n is cut, the maximum revenue of the rod with length ranging from 1 to n-1 will all be considered and the maximum one is returned as the optimal solution of rod with length n. Therefore, this always give the optimal solution.

4) There are two for loop in this program. Time complexity O(n^2)