Date

Name : Muhammad Hamza Khan
Roll : 21L-5654

Task 1:

We can (split) cut the rod of length 6 in following lengths and get prices:

	length of rod	Price
(i)	6	17
(ii)	1 1 1 1 1 1	1+1+1+1+1+1 = 6
(iii)	1 1 1 1 2	1+1+1+1+5 = 9
(iv)	1 1 2 2	1+1+5+5 = 12
(v)	1 1 1 3	3+8 = 11
(vi)	1 1 4	1+1+9 = 11
(vii)	1 2 3	1+5+8 = 14
(viii)	2 2 2	5+5+5 = 15
(ix)	1 5	1+10 = 11
(x)	2 4	5+9 = 14
(xi)	3 3	8+8 = 16
(xii)	4 2	9+5 = 14

The rest will be same as these like another can be 2 & 4 split but split of 2 and 4 or 4 and 2 gives same price i.e 14.

Date

Task 2:

Given rod of length 6. We can make dp array using (with) top down approach.

Given Prices VS

1	5	8	9	10	17	17	20	24	30
1	2	3	4	5	6	7	8	9	10

dp array PS

1	5	8	10	13	~~17~~
1	2	3	4	5	6

Dry run:

$dp[1] = 1$

$dp[2] = \begin{cases} V_1 + P[1] = 1+1 = 2 \\ V_2 = 5 \checkmark \end{cases}$

$dp[3] = \begin{cases} V_1 + P[2] = 1+5 = 6 \\ V_2 + P[1] = 5+1 = 6 \\ V_3 = 8 \checkmark \end{cases}$

$dp[4] = \begin{cases} V_1 + P[3] = 1+8 = 9 \\ V_2 + P[2] = 5+5 = 10 \checkmark \\ V_3 + P[1] = 8+1 = 9 \\ V_4 = 9 \end{cases}$

$dp[5] = \begin{cases} V_1 + P[4] = 1+10 = 11 \\ V_2 + P[3] = 5+8 = 13 \checkmark \\ V_3 + P[2] = 8+5 = 13 \checkmark \\ V_4 + P[1] = 9+1 = 10 \\ V_5 = 17 \end{cases}$

$dp[6] = \begin{cases} V_1 + P[5] = 1+13 = 14 \\ V_2 + P[4] = 5+10 = 15 \\ V_3 + P[3] = 8+8 = 16 \\ V_4 + P[2] = 9+5 = 14 \\ V_5 + P[1] = 10+1 = 11 \\ V_6 = 17 \checkmark \end{cases}$

Date

Task 3:

Algorithm for keeping track of cuts of rod. We declare an array of size n and ~~as~~ initialize it with zero. and then compare with maximum & then ~~with~~ calculate optimal cuts.

```
Rod Cut (Price[], n) {
    dp[0] = 0 ;
    Cut_splits [n];

    for  i = 1  to  n  do
        q = INT_MIN
        for  j = 1  to  i  do
            ~~if max(q,~~
            if (q < price[j] + dp[i-j]) then
                q = price[j] + dp[i-j]

                Cut_splits[i] = j
            end if
        end for
    dp[i] = q
    .
    Optimal cutting [n]
    i=0, ts
    while  n>0  do
            ts = Cut_splits [n]
            Optimal cutting [i] = ts
            n - = ts
            i++
    end while
end for ?
```

return Optimal cutting;