Lab 9. Introduction to algorithms

This is ninth Lab for CS 566. This problem was given in lecture.

Task 1. Solve the problem "Tribonacci Number" from https://leetcode.com/problems/n-th-tribonacci-number/description/ using Python3.

Use the box below, to paste the working code. The format of the code should be identical to LeetCode platform. (4 points)

```
class Solution:
    def tribonacci(self, n: int) -> int:
        t = [0,1,1]
        if n < 3:
            return t[n]

        for i in range(3, n+1):
            t[0], t[1], t[2] = t[1], t[2], sum(t)

        return t[2]
```

Do not modify the testing code below. If you get message "Mistake in test case #", it means that you algorithm is incorrect.

```
#test_case_1
nums = 25
expected = 1389537
actual = Solution().tribonacci(nums)
assert actual==expected, "Mistake in test case 1"
print("OK")
```

```
OK
```

Write analysis of the Memory Complexity and Time Complexity using Aymptotic Notation O. (1 point)

Memory Analysis: O(1)

Time Analysis: O(n)

Task 2. Solve the problem "ROD Cutting" from Course TextBook using Python3.

Use the box below, to paste the working code. This is **not** leetcode, you just need to implement a function and pass the tests (4 points)

```
def rod_cutting(prices, n):
    r = [0 for x in range(n+1)]
    for i in range(1, n+1):
        q = -float('INF')
        for j in range(i):
            q = max(q,prices[j] + r[i-j-1])
        r[i] = q

    return r[n]

# Testing the function
print("Maximum Revenue:", rod_cutting([1, 5, 8, 9, 10, 17, 17, 20], 8))  # Expec
print("Maximum Revenue:", rod_cutting([2, 5, 7, 8], 4))  # Expected: 10
print("Maximum Revenue:", rod_cutting([1, 2, 3, 4, 5], 5))  # Expected: 5
print("Maximum Revenue:", rod_cutting([3, 5, 8, 9], 4))  # Expected: 10
print("Maximum Revenue:", rod_cutting([1, 5, 8, 9, 10, 17, 17, 20, 24, 30], 10))
```

```
Maximum Revenue: 22
Maximum Revenue: 10
Maximum Revenue: 5
Maximum Revenue: 12
Maximum Revenue: 30
```

Write analysis of the Memory Complexity and Time Complexity using Asymptotic Notation O. (1 point)

Memory Analysis: O(n)

Time Analysis: O(n^2)

Task 3. Solve the problem "Coin Change II" from https://leetcode.com/problems/coin-change-ii/description/ using Python3.

Use the box below, to paste the working code. The format of the code should be identical to LeetCode platform. (4 points)

```
from typing import List
class Solution:
    def change(self, amount: int, coins: List[int]) -> int:
        dp = [0] * (amount + 1)
        dp[0] = 1
        for coin in coins:
            for j in range(coin, amount + 1):
                dp[j] += dp[j - coin]
        return dp[amount]
```

```
#test_case_1
amount = 5
coins = [1,2,5]
expected = 4
actual = Solution().change(amount, coins)
assert actual==expected, "Mistake in test case 1"
print('OK')
```

```
OK
```

Write analysis of the Memory Complexity and Time Complexity using Asymptotic Notation O. (1 point)

Memory Analysis: O(m), m is amount

Time Analysis: O(n*m), n is coin, m is amount