# HW 10. Introduction to algorithms

This is tenth homework for CS 566.

## Task 1. Solve the problem "Trapping Rain Water" from [https://leetcode.com/problems/trapping-rain-water/description/](https://leetcode.com/problems/trapping-rain-water/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)

```
import sys
from typing import List

class Solution:
    def trap(self, height: List[int]) -> int:
        n = len(height)
        # max height to left and right of current index
        max_left = [0] * n
        max_right = [0] * n
        max_left[0] = height[0]
        max_right[n-1] = height[n-1]

        for i in range(1, n):
            max_left[i] = max(height[i], max_left[i-1])

        for i in range(n-2, -1, -1):
            max_right[i] = max(height[i], max_right[i+1])

        num_water = 0
        for i in range(0,n):
            num_water += min(max_left[i], max_right[i]) - height[i]

        return num_water
```

## 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
height = [0,1,0,2,1,0,1,3,2,1,2,1]
actual = Solution().trap(height)
expected = 6
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(n)

Time Analysis: O(n)

## Task 2. Solve the problem "Longest Palyndromic Substring" from https://leetcode.com/problems/longest-palindromic-substring/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 longestPalindrome(self, s: str) -> str:
        # dp approach
        ans = ""
        count = 0
        n = len(s)

        dp = [[False]*n for i in range(n)]

        for i in range(n):
            for j in range(n-i):
                k = i+j
                if i == 0:
                    dp[j][k] = True
                elif i == 1:
                    is_palindrome = (s[j] == s[k])
                    dp[j][k] = is_palindrome
                else:
                    is_palindrome = (s[j] == s[k] and dp[j+1][k-1])
                    dp[j][k] = is_palindrome
                if dp[j][k] and count < k-j+1:
                    ans = s[j:k+1]
                    count = len(ans)

        return ans
```

```
s = "babad"
expected = "bab"
actual = Solution().longestPalindrome(s)
```

```
assert expected == actual, "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(n^2)

Time Analysis: O(n^2)

## Task 3. Solve the problem "Gas Station" from [https://leetcode.com/problems/gas-station/description/](https://leetcode.com/problems/gas-station/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 canCompleteCircuit(self, gas: List[int], cost: List[int]) -> int:
        # greedy
        if sum(gas) < sum(cost):
            return -1

        total = 0
        starting_pos = 0
        for i in range(len(gas)):
            total += (gas[i] - cost[i])

            # if total negative, the position doesn't work so reset position
            if total < 0:
                total = 0
                starting_pos = i + 1

        return starting_pos
```

```
gas = [1,2,3,4,5]
cost = [3,4,5,1,2]
expected = 3
actual = Solution().canCompleteCircuit(gas, cost)
assert expected==actual, "Mistake in case 1"
print("OK")
```

```
OK
```

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

Memory Analysis: O(1)

Time Analysis: O(n)