Lab 5. Introduction to algorithms

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

Task 1. Solve the problem "Binary Search" from https://leetcode.com/problems/binary-search/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 search(self, nums: List[int], target: int) -> int:
        left, right = 0, len(nums)-1
        while left<=right:
            mid = left+(right-left)//2
            if nums[mid] == target:
                return mid
            elif target>nums[mid]:
                left = mid+1
            else:
                right = mid - 1
        return -1
```

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 = [-1,0,3,5,9,12]
target, expected = 9, 4
actual = Solution().search(nums, target)
assert actual==expected, "Mistake in test case 1"

#test_case_2
nums = [-1,0,3,5,9,12]
target, expected = 11, -1
actual = Solution().search(nums, target)
assert actual==expected, "Mistake in test case 2"
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(logn)

Task 2. Solve the problem "Find minimum in rotated sorted array" from https://leetcode.com/problems/find-minimum-in-rotated-sorted-array/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 findMin(self, nums: List[int]) -> int:
        left, right = 0, len(nums)-1
        while left<right:
            mid = left+(right-left)//2
            if nums[mid]>nums[right]:
                left = mid + 1
            else:
                right = mid
        return nums[left]
```

```
#test_case_1
expected, nums = 1, [3,4,5,1,2]
actual = Solution().findMin(nums)
assert expected == actual, "Mistake in test case 1"

expected, nums = 0, [4,5,6,7,0,1,2]
actual = Solution().findMin(nums)
assert expected==actual, "Mistake in test case 2"
expected, nums = 11, [11,13,15,17]
actual = Solution().findMin(nums)
assert expected==actual, "Mistake in test case 3"

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(logn)

Task 3. Solve the problem "Search 2D Matrix" from https://leetcode.com/problems/search-a-2d-matrix/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 searchMatrix(self, matrix: List[List[int]], target: int) -> bool:
        num_rows, num_cols = len(matrix), len(matrix[0])
        # binary search to find if there is a row that is within target range
        low, high = 0, num_rows - 1
        target_row = 0
        while low<=high:
            mid_row = low + (high-low)//2
            target_row = mid_row
            if target>matrix[mid_row][-1]:
                low = mid_row + 1
            elif target<matrix[mid_row][0]:
                high = mid_row - 1
            else:
                break
        # if none of rows are within range, return false
        if low>high:
            return False

        # binary search to find if the value exist in the row that the target f
        left, right = 0, num_cols - 1
        while left<=right:
            mid = left + (right-left)//2
            x = matrix[target_row][mid]
            if target == x:
                return True
            elif target>x:
                left = mid + 1
            elif target<x:
                right = mid - 1
        return False
```

```
#test_case_1
expected, nums, target = True, [[1,3,5,7],[10,11,16,20],[23,30,34,60]], 3
actual = Solution().searchMatrix(nums, target)
assert expected == actual, "Mistake in test case 1"
```

```
expected, nums, target = False, [[1,3,5,7],[10,11,16,20],[23,30,34,60]], 13
actual = Solution().searchMatrix(nums, target)
assert expected==actual, "Mistake in test case 2"

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(log(n*m))