**📝 Infinite Champions Programme – Day 7 (Assignment Sheet)**

**📌 Instructions  
• Deadline: Submit your solutions by 4th October, 2025, EOD.  
• Platform: Test your solutions on LeetCode  
• Collaboration: Discussing concepts is encouraged, but all code must be your own.**

1. [**House Robber (198)**](https://leetcode.com/problems/house-robber/)  
   • Problem: You are a professional robber planning to rob houses along a street. Each house has some money, but adjacent houses cannot be robbed on the same night.  
   • Objective: Use dynamic programming to maximize the total amount of money robbed.  
   • YouTube Solution (Java): [House Robber – Java Solution](https://www.youtube.com/watch?v=73r3KWiEvyk)

class Solution {

public int rob(int[] nums) {

int n = nums.length;

Integer[] m = new Integer[n];

return solve(nums, n - 1, m);

}

public int solve(int[] nums,int i, Integer[] m){

if(i<0){

return 0;

}

if(i==0){

return nums[i];

}

if (m[i] != null) return m[i];

int rob = nums[i] + solve(nums, i - 2, m);

int skip = solve(nums, i - 1, m);

return m[i] = Math.max(rob, skip);

}

}

1. [**Maximum Subarray (53)**](https://leetcode.com/problems/maximum-subarray/)  
   • Problem: Given an integer array nums, find the contiguous subarray with the largest sum.  
   • Objective: Apply Kadane’s algorithm (DP-based) to find the maximum subarray sum.  
   • YouTube Solution (Java): [Maximum Subarray – Java Solution](https://www.youtube.com/watch?v=5WZl3MMT0Eg)

class Solution {

    public int maxSubArray(int[] nums) {

        int n=nums.length;

        int currS = nums[0];

        int maxS = nums[0];

        for (int i = 1; i < n; i++) {

            currS = Math.max(nums[i], currS + nums[i]);

            maxS = Math.max(maxS, currS);

        }

        return maxS;

    }

}

1. [**Unique Paths (62)**](https://leetcode.com/problems/unique-paths/)  
   • Problem: A robot is at the top-left corner of an m x n grid. It can only move right or down. Find the number of unique paths to reach the bottom-right corner.  
   • Objective: Use dynamic programming with grid traversal to count possible paths.  
   • YouTube Solution (Java): [Unique Paths – Java Solution](https://www.youtube.com/watch?v=IlEsdxuD4lY)

class Solution {

public int uniquePaths(int m, int n) {

long res = 1;

int total = m + n - 2;

int r = Math.min(m-1, n-1);

for (int i = 1; i <= r; i++) {

res = res \* (total - r + i) / i;

}

return (int) res;

}

}

**📚 Submission Checklist  
• Time and space complexity analysis for each solution.  
• Test cases demonstrating the correctness of your solutions.**