Module - 21.5: Practice Problems

All the problems below are related to Dynamic Programming. You have to solve each problem using both **memoization** and **tabulation** methods.

1. Write code to output the n-th **Tribonacci** number. A Tribonacci series is a series where each number of the series is the sum of **previous three numbers (**except the 1st three numbers which are all 1-s)

   The first few numbers of the series is as follows:

   `1, 1, 1, 3, 5, 9, 17, 31, 57, 105, 193, 355, 653, 1201 . . .`

   What is the **time** and **space** complexity of your code?

2. Read the problem: https://leetcode.com/problems/climbing-stairs/
   Now,
   a. Define your DP state
   b. Write the recurrence relation
   c. Write the base case
   d. Write the program in C++
   e. Write the **time** and **space** complexity of your code

3. Do the same as **problem-2** for the following problem:
   https://leetcode.com/problems/min-cost-climbing-stairs/

4. Do the same as **problem-2** for the following problem:
   https://leetcode.com/problems/house-robber/

5. We solved the problem Frog 1 in our theory module where we saw that a frog can either go to the `(i+1)` or the `(i+2)` -th stone from stone `i.`

   What if we changed the problem such that the frog can now jump to the `(i+1)` or the `(i-1)` -th stone from stone `i`.

   Can we now solve the problem with dynamic programming? Why or why not?

   Can you solve this using graphs? **(No code is required)**