1. **Atcoder Task B :** Frog 2
   TASK - B : Frog 2 = *https://atcoder.jp/contests/dp/tasks/dp_b*
   For this case, here we can jump at most k stone, so time complexity wil be O(n * k) which was before only O(n).

2. **Atcoder Task B :** Frog 2 in Code.
   Solving Frog2 with Memoizatin + Tabulation in both method.

3. **Solving DP with 2 dimensions : 2Dimension DP.**
   TASK - C : Vacation : *https://atcoder.jp/contests/dp/tasks/dp_c*

   nth day (x, y, d) happiness :
   fun(n, x) = max { fun(n-1, without x) + happiness (n, x) }

   If we cann't solve a dp problem with only one state, we will try to solve that problem with more than one state.

   Leap of Faith = Backtracking

   Time Complexity : O (n*x) if, fun(a, b, c, d) then → O(a*b*c*d)
   If fun(a, b, c, d):
   for (k - length) :
   for(k - length)
   Then → O(a*b*c*d*k^2)

   For this problem,
   Time Complexity : O(n*x*3) → O(9n) → **O(n)**
   Space Complexity : dp[n][x] : O(n * x) → O(3n) → **O(n)**

4. **CODE : Implementation in Code in Both Memoization + Tabulation Method**
   TASK - C : Vacation : *https://atcoder.jp/contests/dp/tasks/dp_c*