3) 16 pts

You have found a hidden treasure that contains *n* diamonds placed in a row. You know each of these diamonds' values [v_0, v_1, …, v_{n-1}]. Ideally, you would have taken all the diamonds, but this treasure is cursed, and the cave will collapse if you pick up 3 consecutive diamonds. Knowing this, find the maximum total value of the diamonds you can pick up such that no three consecutive diamonds are picked.

a) Define (in plain English) the subproblems to be solved. (4 pts)

dp[i] = Max possible total value from v_0 to v_i, such that no three elements are consecutive.

b) Write a recurrence relation for the subproblems (4 pts)

dp[i] = Max of the following 3 cases:

- dp[i] = dp[i-1] (Exclude v_i)
- dp[i] = dp[i-2] + v_i (Exclude v_{i-1})
- dp[i] = dp[i-3] + v_i + v_{i-1} (Exclude v_{i-2})

Rubric: one point for each of the 3 cases, one for taking max of the 3

c) (6 pts for part c)
Using the recurrence formula in part b, write pseudocode using iteration to find the maximum total value of diamonds you can obtain without having the cave collapse. (4 pts)
Make sure you specify base cases and their values (2 pts)

Base case:

- dp[0] = v_0
- dp[1] = $v_0 + v_1$
- dp[2] = max(dp[1], $v_1 + v_2$, $v_0 + v_2$)
    - Alternatively, dp[-1] = 0

Pseudocode:

```
*Compute Base cases above*
For i=3 to n-1, do:
        dp[i] = max of the 3 cases              \\ refer to the recurrence relation

return dp[n-1]
```

Rubric:

- 2 points for the base cases
- Pseudocode: 1 point for the order/structure of the computations, 2 points for the loop, 1 point for the final return

d) What is the time complexity of your solution? Is your algorithm efficient? (2 pts)

O(n). Thus, it is efficient.