Generated Final Exam Question

One creative recursion-related exam question could be:

Question X:

Imagine a kingdom where every year, one gold coin is passed to the next generation. However, if a generation has multiple children, they each receive an exact copy of the first coin while the original is retained by their parent. This tradition, like a "golden recursion", has been practised since generations.

We are interested to know the total number of gold coins accumulated by the kingdom over the years. You are provided with a list of generations and the number of children each generation had. The first element of the list is the first generation.

Write a recursive function `calculate_coins(generations: List[int]) -> int` that takes a list of integers (1 <= `generations[i]` <= 5 for all `i`) as input representing the number of children each generation had and returns the total number of gold coins up to (and including) the current generation.

For example,

`calculate_coins([2, 1, 3])` should return `8`, because:

1st generation: 1 gold coin is passed to 2 children: 3 gold coins exist now.

2nd generation: Those 2 gold copies are each passed onto 1 child: 5 gold coins exist now.

3rd generation: Those 2 gold copies are each passed onto 3 children: 8 gold coins exist now.

```python
def calculate_coins(generations):
```

```
    """
    >>> calculate_coins([2])
    3

    >>> calculate_coins([3, 5, 2])
    63

    >>> calculate_coins([1, 1, 1, 1, 1])
    5
    """
    #your code here
```

Note that you should not use any loops or external libraries. This function must be solved solely with the use of recursion. The list `generations` will always have at least one element, and all the elements will be positive integers.