# Recursion: exam-level questions

If you need help reviewing Recursion, take a look at these resources:

- Albert's and Robert's slides (recursion) (https://docs.google.com/presentation/d/10GYJwCNS0N41pTcUWxNpLlBpFpCxzBG3OVSKjkHLopQ/edit)
- Albert's and Robert's slides (tree recursion) (https://docs.google.com/presentation/d/1PgyLb9WUgGpnjX_—cjtNHO3iXT-aHQW8O2RKGPc_IE/edit#slide=id.p)

Each question has a "Toggle Solution" button -- click it to reveal that question's solution.

## Code-Writing Questions

### Question 1

In game theory, a *subtraction game* is a simple game with two players, player 0 and player 1. At the beginning, there is a pile of *n* cookies. The players alternate turns; each turn, a player can take anywhere from 1 to 3 cookies. The player who takes the last cookie wins. Fill in the function `can_win`, which returns `True` if it is possible to win starting at the given number of cookies. It uses the following ideas:

- if the number of cookies is negative, it is impossible to win.
- otherwise, the current player can choose to take either 1, 2, or 3 cookies.
- evaluate each action: if that action forces the opponent to lose, then return True (since we can win)
- if none of the actions can force a win, then we can't guarantee a win.

```
def can_win(number):
    """Returns True if the current player is guaranteed a win
    starting from the given state. It is impossible to win a game
    from an invalid game state.

    >>> can_win (-1) # invalid game state
    False
    >>> can_win (3) # take all three !
    True
    >>> can_win (4)
    False
    """
    "*** YOUR CODE HERE ***"
```

Toggle Solution

## Environment Diagrams

### Question 2

```
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n-1)

ans = factorial(2)
```

Toggle Solution

## Question 3

```
def foo(n):
    i = 0
    if n == 0:
        return 0
    result = foo(n - 2)
    i += 1
    return i + result

result = foo(4)
```

Toggle Solution

## Question 4

```
def bar(f):
    def g(x):
        if x == 1:
            return f(x)
        else:
            return f(x) + g(x - 1)
    return g

f = 4
bar(lambda x: x + f)(2)
```

Toggle Solution