The prime numbers are not regularly spaced. For example from `2` to `3` the gap is `1`. From `3` to `5` the gap is `2`. From `7` to `11` it is `4`. Between 2 and 50 we have the following pairs of 2-gaps primes: `3-5, 5-7, 11-13, 17-19, 29-31, 41-43`

A prime gap of length n is a run of n-1 consecutive composite numbers between two **successive** primes (see: http://mathworld.wolfram.com/PrimeGaps.html).

We will write a function gap with parameters:

`g` (integer >= 2) which indicates the gap we are looking for

`m` (integer > 2) which gives the start of the search (m inclusive)

`n` (integer >= m) which gives the end of the search (n inclusive)

In the example above `gap(2, 3, 50)` will return `[3, 5] or (3, 5) or {3, 5}` which is the first pair between 3 and 50 with a 2-gap.

So this function should return the **first** pair of two prime numbers spaced with a gap of `g` between the limits `m`, `n` if these numbers exist otherwise `nil or null or None or Nothing` (depending on the language).

In C++ return in such a case `{0, 0}`. In F# return `[||]`. In Kotlin return `[]`

#Examples: `gap(2, 5, 7) --> [5, 7] or (5, 7) or {5, 7}`

`gap(2, 5, 5) --> nil. In C++ {0, 0}. In F# [||]. In Kotlin return []`

`gap(4, 130, 200) --> [163, 167] or (163, 167) or {163, 167}`

([193, 197] is also such a 4-gap primes between 130 and 200 but it's not the first pair)

`gap(6,100,110) --> nil or {0, 0}` : between 100 and 110 we have `101, 103, 107, 109` but `101-107` is not a 6-gap because there is `103` in between and `103-109` is not a 6-gap because there is `107` in between.