task given a finite set X, a function $f : X \longrightarrow X$ and an element $x \in X$ compute the periodicity parameters of the sequence $x, f(x), f(f(x)), f(f(f(x))), \ldots$

clarification one must compute the minimal $s \geq 0$ such that $f^s(x)$ appears infinitely often in the sequence (the number of steps to periodicity), as well as the minimal $\ell \geq 1$ such that $f^{s+\ell}(x) = f^s(x)$ (the period).

algorithm (Floyd) first find the minimum $k \geq 1$ for which $f^k(x) = f^{2k}(x)$. then, the first $r \geq 0$ occurence of $f^r(x) = f^{k+r}(x)$ is the number of steps to enter the infinite loop. finally, the first $p \geq 1$ such that $f^r(x) = f^{r+p}(x)$ is the length of the loop.

pseudo code

```
def compute_periodicity_parameters(f,x):
    slow = f(x)
    fast = f(f(x))
    while slow != fast:
       slow = f(slow)
       fast = f(f(fast))

    searcher = x
    no_steps_to_cycle = 0
    while searcher != slow:
       no_steps_to_cycle += 1
       searcher = f(searcher)
       slow = f(slow)

    loop_entry = searcher
    searcher = f(searcher)
    cycle_length = 1
    while searcher != loop_entry:
       cycle_length += 1
       searcher = f(searcher)

return no_steps_to_cycle, cycle_length
```

exercise verify the algorithm is correct and of linear time complexity `O(no_steps_to_cycle + cycle_length)`.

algorithm (Brent) let $0, 1, 3, 7, 15, \ldots$ be the comparison points. find the first $k \geq 1$ where $f^k(x) = f^m(x)$ and m is the maximal comparison point prior to k. then $k - m$ is the length of the period.

pseudo code

```
def compute_periodicity_parameters(f,x):
   stopping_index = 0, stopping_node = x
   travelling_index = 1, travelling_node = f(x)
   while travelling_node != stopping_node:
      if travelling_index == 2*stopping_index + 1:
         stopping_index = travelling_index, stopping_node = travelling_node
      travelling_index += 1, travelling_node = f(travelling_node)

   cycle_length = travelling_index - stopping_index

   no_steps_to_cycle = 0, tail = x, head = f^(cycle_length)(x)
   while head != tail:
      no_steps_to_cycle +=1, head = f(head), tail = f(tail)

return no_steps_to_cycle, cycle_length
```

exercise verify the algorithm is correct and of linear time complexity `O(no_steps_to_cycle + cycle_length)`.