3. Problem 3

```
Algorithm findElementEqualToItsIndex (A, start, end)                Count of operations
Input: sorted array A, starting position start, ending position end
Output: true if element A[m] = m is found, false otherwise
mid = (start + end) / 2                                                      3
if (A[mid] = mid) then                                                       3
    return true                                                              1
if (A[mid] < mid and start != end) then                                      3
    return findElementEqualToItsIndex (A, mid + 1, end)                  3 + T(n/2)
if (A[mid] > mid and start != end) then                                      3
    return findElementEqualToItsIndex (A, start, mid)                    2 + T(n/2)
return false                                                                 1
```

$$T(n) = \begin{cases} 7 & if\ n = 1 \\ T\left(\frac{n}{2}\right) + 10 & otherwise \end{cases}$$

$a = 1,\ \ b = 2,\ \ c = 10,\ \ d = 7,\qquad k = 0$

$a = 1 = b^k = 2^0 = 1 \rightarrow from\ the\ master\ formula: T(n)\ is\ \Theta\left(n^k \log n\right)$

$\therefore T(n)\ is\ \mathrm{o}(\log n),\ \ since\ all \log n\ functions\ are\ o(n).$