## Exercise 1

Consider the pseudo-code for FIND-ELEMENT:

```
FIND-ELEMENT(A, a)
1  j = 0
2  for i = 1 to A.length
3       if A[i] = a
4            j = i
5  return j
```

1. Step *meticulously* through the algorithm for each of the following inputs. Note down each line that is executed, what the value of $i$ and $j$ is and what is returned in the end.
   - ('m', 'f', 'a', 'b', 'k'), 'b'
   - (7, 2, 1, 4), 7
   - (), 'd'
   - (0, 1, 0, 1, 0, 1, 0), 1
   - (3, 4, 2, 4), 4
   - ('p', 'x', 'f', 'l'), 'm'

Consider the pseudo-code for FIND-ELEMENT-V2, an alternative version the algorithm:

```
FIND-ELEMENT-V2(A, a)
1  i = A.length
2  while i > 0
3       if A[i] = a
4            j = i
5       i = i − 1
6  return j
```

2. Does FIND-ELEMENT and FIND-ELEMENT-V2 solve the same problem? Step through the algorithm to convince yourself!

Consider the pseudo-code for FIND-ELEMENT-V3, an alternative version the algorithm:

```
FIND-ELEMENT-V3(A, a)
1  i = A.length
2  while i > 0 and A[i] ≠ a
3       i = i − 1
4  return i
```

3. Does FIND-ELEMENT-V3 solve the same problem as either or both of the two previous algorithms? Once again, journey through the tedious process of stepping through the algorithm to convince yourself!

## Exercise 2

1. Write pseudo-code for an algorithm COUNT-INSTANCES, which takes as input a sequence $A$ of characters and a character $a$ and outputs the number of occurrences of $a$ in $A$. So, for the input pair (('a', 'b', 'b', 'a'), 'a') the output should be 2.

2. Write pseudo-code for an algorithm CHECK-EQUALITY, which takes as input two sequences $A$ and $B$ of characters and outputs TRUE if $A$ and $B$ are identical and FALSE otherwise. NB: Note that it is not required that the two input sequences are of the same length (your algorithm needs to detect this)!