(b) [**4 marks**] Consider the following function.

```
def f2(numbers: list[int], m: int) -> None:
    """Precondition: m >= 0"""
    for i in range(0, m):
        if i in numbers:
            print('Found')
        else:
            print('Not found')
```

Perform an **upper bound analysis of the worst-case running time** of this function, in terms of n, the length of `numbers`, and the input value m. The Big-O expression that you conclude should be *tight*, meaning that the worst-case running time should be Theta of this expression, but you are not required to show that. Use "at most" or $\leq$ to explicitly indicate where a step count expression is an upper bound.

The inner if loop runs a maximum of n iterations, each iteration takes one step, thus the inner if loops runs at most n steps. The outer for loop runs exactly m iterations, each iteration takes at most n steps, thus the function takes at most m.n steps to run.

Thus $WC_{f_2} \in O(m.n)$

Lets say numbers is a list of n consecutive integers starting at m. For any i in range (0, m), i will not be found in list numbers, causing the inner if loop to run n iterations of 1 step always. The outer for loop runs m iterations for n steps each. Thus $WC_{f_2} \in \Omega(m.n)$.

Thus $WC_{f_2} \in \Theta(m.n)$