Algorithm Theory: Exercise Sheet 1
Kai Malaka (km324), Manuel Di Biase (md509)

Exercise 3

a).

```
FUNCTION maximum_area_rectangle( array )
    maximum_area = 0
    FOR i = 0 to array length - 1
        length = 0
        height = array[i]
        FOR j = i to array length - 1
            length = length + 1
            height = min{height, array[j]}
            maximum_area = max{maximum_area, length * height}
        END FOR
    END FOR
    RETURN maximum_area
END FUNCTION
```

b).

```
FUNCTION maximum_area_rectangle_intersecting_bar( array , bar )
    left = bar - 1
    right = bar + 1
    height = array[bar]
    maximum_area = height
    FOR length = 2 to array length
        IF right >= array length or (left >= 0 and array[left] > array[right]):
            height = min{height, array[left]}
            left = left - 1
        ELSE
            height = min{height, array[right]}
            right = right + 1
        END IF
        maximum_area = max{maximum_area, length * height}
    END FOR
    RETURN maximum_area
END FUNCTION
```

The running time of this function consists of three parts:

(1) Initialization:
The assignment of values to the variables left, right, height, and maximum_area takes a constant time. - c_1

(2) Amount of iterations in the main loop:
The loop always runs from 2 to the array length. For an array of size n, this corresponds to n-1 iterations.

(3) Code in the loop:
The condition takes a constant time to evaluate, because it is only using comparison operators. The code in both paths only uses a min function and one addition/subtraction which also takes a constant time to execute. After the condition is a max function, which also takes a constant time to evaluate. So the whole code block inside the loop takes a constant time to evaluate. - c_2

The running time of the function is $c_1 + (n-1) \cdot c_2$ which is a linear function, so it is in $\mathcal{O}(n)$

c).

```
FUNCTION maximum_area_rectangle(array: list[int]) -> int:
    IF length of array == 1:
        RETURN array[0]
    END IF
    RETURN max{
        maximum_area_rectangle( lower half of array ),
        maximum_area_rectangle_intersecting_bar(array, central array element),
        maximum_area_rectangle( upper half of array ) }
END FUNCTION
```

maximum_area_rectangle_intersecting_bar is the function defined above. If the array doesn't have a clear center element either one of the two central elements can be used.

Running timen $T(n)$:
Basis:
let $n = 1$.
The evaluation of the condition and the return statement happen in some constant time c_1.
$T(1) = c_1$

let $n \geq 2$.
The evaluation of the condition and the max function executes in some constant time c_2.
The evaluation of the maximum_area_rectangle_intersecting_bar is, as shown above in $\mathcal{O}(n)$
Lastly the maximum_area_rectangle function gets called twice with arrays of size $n/2$
$T(n) = 2 \cdot T(\frac{n}{2}) + c_2 + n \cdot c_3$

For this recurrence relation the master theorem states that $T(n) = \mathcal{O}(n \; log \; n)$