Assignment 9

I

```
lower <- 0
upper <- length(A)

while (A[lower + (upper-lower)/2] is not i)
        if (A[lower + (upper-lower)/2] < i)
                do lower <- lower + (upper-lower)/2
        else
                do upper <- lower + (upper-lower)/2

return lower + (upper-lower)/2
```

This algorithm is correct because it will half the array at every iteration. When the middle of the array is smaller than i, the lower bound will jump tot he middle of the array. The opposite happends when the array element is larger than i. This way the algorithm will pinpoint the element with value i.

II

1.

Marieke

Guess: O(2^n)
Induction: n = 1 -> 2^n = 2 = T(1)
Substitution:

T(n) = 2(T(n-1)) + 1
= 2 (2^n-1) + 1
= 2^n + 1
= O(2^n)

Joost:

David

Guess: O(5*log(n))
Induction: n = 1 -> 5*log(n) + n = 1 = T(1)

Substitution:

T(n) = 5*log(n) + n

III

max-income (lower, upper, I)

if (lower == upper)
    return I[lower]
else
    income = $\sum_{k=0}^{n} I[k]$
    if (income < max-income(lower, upper-1, I)
        income <- max-income(lower, upper-1, I)
    if (income < max-income(lower+1, upper, I)
        income <- max-income(lower+1, upper, I)
    return income

Complexity: O(2^n)

For every recursive call, the function is executed twice.

IV

Steps:

1. If n = 1, add one block to the left-bottom corner.
2. Else, devide the total space into four equal spaces, namely the left-top, right-top, left-bottom and right-bottom. Recursively solve these spaces. After solving, turn the left-top corner 90 degrees to the right and the right-bottom corner 90 degrees to the left. Continue with step 3.
3. Add one block in the middle of the created image.

Complexity: