Problem 2.3

Correctness of Horner's rule

> The following code fragment implements Horner's rule for evaluating a polynomial
>
> $$P(x) = \sum_{k=0}^{n} a_k x^k = \\ = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + xa_n)\cdots))$$
>
> given the coefficients $a_0, a_1, \ldots, a_n$ and a value for x:
>
> ```
> y = 0
> for i = n downto 0
>     y = aᵢ + x·y
> ```
>
> 1. In terms of Θ-notation, what is the running time of this code fragment for Horner's rule?
> 2. Write pseudocode to implement the naive polynomial-evaluation algorithm that computes each term of the polynomial from scratch. What is the running time of this algorithm? How does it compare to Horner's rule?
> 3. Consider the following loop invariant:
>
>   > At the start of each iteration of the **for** loop of lines 2-3,
>   >
>   > $$y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k$$
>   >
>   > Interpret a summation with no terms as equaling 0. Following the structure of the loop invariant proof presented in this chapter, use this loop invariant to show that, at termination, $y = \sum_{k=0}^{n} a_k x^k$.
>
> 4. Conclude by arguing that the given code fragment correctly evaluates a polynomial characterized by the coefficients $a_0, a_1, \ldots, a_n$.

1. Running time

Obviously $\Theta(n)$.

2. Naive algorithm

We assume that we have no exponentiation in the language. Thus:

```
y = 0
for i = 0 to n
    m = 1
    for k = 1 to i
        m = m·x
    y = y + aᵢ·m
```

The running time is $\Theta(n^2)$, because of the nested loop. It is, obviosly, slower.

3. The loop invariant

Initialization: It is pretty trivial, since the summation has no terms, which implies $y = 0$.

Maintenance: By using the loop invariant, in the end of the i-th iteration, we have:

$$y = a_i + x \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k = a_i x^0 + \sum_{k=0}^{n-i-1} a_{k+i+1} x^{k+1} = \\ = a_i x^0 \sum_{k=1}^{n-i} a_{k+i} x^k = \sum_{k=0}^{n-i} a_{k+i} x^k$$

Termination: The loop terminates at $i = -1$. If we substitute, we get:

$$y = \sum_{k=0}^{n-i-1} a_{k+i+1} x^k = \sum_{k=0}^{n} a_k x^k$$

4. Conclude

It should be fairly obvious, but the invariant of the loop is a sum that equals a polynomial with the given coefficients.