a)

Schleifeninvariante:

Für $n \geq i \geq 0$ gilt vor dem i-ten Durchlauf:

$$y = \sum_{j=0}^{i-1} x[n-j] \cdot a^{i-1-j}$$

Initialisierung:

Vor dem ersten (d.h. i = 0) Durchlauf gilt:

$$y = \sum_{j=0}^{-1} x[n-j] \cdot a^{0-1-j} = 0$$

, nach Definition des Summenzeichens.

Dies ist erfüllt. Siehe Zeile 2:

y := 0;

Aufrechterhaltung:

Wenn die Invariante vor dem i-ten Schritt galt, dann folgt im i-ten Schritt:

$$y = x[n-i] + a \cdot \sum_{j=0}^{i-1} x[n-j] \cdot a^{i-1-j} = x[n-i] + \sum_{j=0}^{i-1} x[n-j] \cdot a^{i-j}$$

$$= x[n-i] \cdot a^{i-i} + \sum_{j=0}^{i-1} x[n-j] \cdot a^{i-j} = \sum_{j=0}^{i} x[n-j] \cdot a^{i-j} = \sum_{j=0}^{(i+1)-1} x[n-j] \cdot a^{(i+1)-1-j}$$

Dies ergibt also die Invariante vor dem (i+1)-ten Durchlauf.

Terminierung:

Da es sich um eine for-Schleife, bei der die Zählervariable nicht verändert wird, terminiert der Algorithmus nach n+1 Schritten. (von 0 bis einschließlich n).

b)

Pseudocode (nach Vorlesung ohne end 'Befehle'. Blöcke sind an der Rückung zu erkennen)

```
function EVALUATEPOLYNOMIALNAIVE (Real a, Real[] x)
        Setze y:= 0;
        for i := 0 to n do
                y := y + (x[i] * POW(a, i));
        return y;

function POW(Real a, Integer i)
        for j := 0 to i do
                a := a * a;
        return a;
```

Laufzeitbestimmungen:

$POW(Real\ a, Integer\ i) \in O(n)$, da i Schleifendurchläufe mit Konstanter Laufzeit (eine Multiplikation) und anschließend ein weiterer konstanter Befehl (return) ausgeführt werden.

Kostengleichung für EVALUATEPOLYNOMIALNAIVE(Real a, Real[] x) mit n:= LENGTH(x).

$$T(n) \leq \underset{\substack{\uparrow \\ Zeilen\ 2,5}}{2} + \underset{\substack{\uparrow \\ 3,4}}{(n+1)} \cdot n = n^2 + n + 2$$

=> $T(n) \in O(n^2)$

Kostengleichung für EVALUATEPOLINOMIAL(Real a, Real[] x) mit n:= LENGTH(x).

$$T(n) \leq \underset{\substack{\uparrow \\ Zeilen\ 2,6}}{2} + \underset{\substack{\uparrow \\ 3,4,5}}{(n+1)} \cdot (1+1+1) = 3n+5$$

Zeilen 2,6: konstante Werzuweisung

Zeilen 3,4,5: for-Schleife mit n-Durchläufe zu je drei konstanten Kosten (Multiplikation, Addition, Zuweisung).

=> $T(n) \in O(n)$