1a) $f(x, \vec{a}, n) := \sum_{i=0}^{n} a_i \cdot x^i$ 2/3

b) $g_i(x, \vec{a}) := \begin{cases} a_n & , \text{falls } i = 0 \\ a_i + g(x, a_{i-1}) * x & , \text{falls } i > 0 \end{cases}$ 1/3

← würde weiter zu Rekursion führen

c) $f(x, \vec{a}, n)$ führt $n+2$ Additionen und ✓

$\sum_{i=0}^{n+2} i$ Multiplikationen aus. $= \frac{n \cdot (n+1)}{2} + (n+1)$ ✓

$g_i(x, \vec{a})$ führt $n+1$ viele Additionen und ✓

$n+1$ viele Multiplikationen aus ✓ 2/3

d)
```
double modf ( double x, double a[], const int n){
    double array[n];        double f = 0.0;
    double pow = 1;
    for ( int j=0 ; j <= n ; j++){
        pow *= x;
        array[j] = pow;
    }
    for (int i= 0; i<=n ; i++){
        double pow = 1;
        pow = array[i];
        f += a[i] * pow;
    }
```

← array[0] = 5 müsste aber 1 sein

fast! sind ja $2n+2$ 2/3

e) f und g geben beide den gleichen Wert zurück. g benötigt aber weniger Multiplikationen, weshalb sich g eher eignet, wenn man die Anzahl der Multiplikationen gering halten will. ✓ 1/3

$f(n) = g(n)$ Umformung fehlt

2) Die Funktion führt immer eine Addition aus, wenn der Wert für j kleiner ist als der Wert für i geteilt durch n.

Für n=0 und n=1 werden also keine Additionen ausgeführt

Ansonsten werden (n/1)-1 + (n/2)-1 +...+ (n/n)-1 Additionen ausgeführt.

Die Anzahl der Additionen lässt sich als folgende Funktion darstellen:

$$f(n) := \begin{cases} 0, \text{falls } n<2 \\ \sum_{i=2}^{n} (n/i)-1 \text{ , sonst.} \end{cases}$$

Im Extremfall wird die zweite Schleife genau so oft durchlaufen wie die erste. Die erste Schleife wird n mal aufgerufen ($O(n)$). Nun kommt die zweite Schleife dazu, die n-mal aufgerufen wird (Im Extremfall). $\Rightarrow O(n^2)$.

f ✓

logarithmisch

3,5/5

3a) Man kann Nullen und Einsen auf $2^8 = 256$ Weisen anordnen. Ein char kann also 256 verschiedene Zeichen darstellen. ✓ 1

b) Die Großbuchstaben liegen bei den ASCII-Codes 65 bis einschließlich 90. ✓ 1

c) Die Kleinbuchstaben liegen bei den ASCII-Codes 97 bis einschließlich 122. ✓ 1

d) Die Zahlen von 0 bis 9 liegen bei den ASCII-Codes 48 bis einschließlich 57. ✓ 1

e) Der erste Code ist decodiert: "Ich lerne Informatik" ✓
Der zweite Code ist decodiert: "0124689.1" ✓ 2

f)*
```
bool IsNumber(const char array[]){
    bool num = true
    for (int i=0; i<=strlen(a)-1; i++){
        if(a[i] != 48 && a[i] != 49 && a[i]!=50
            && a[i] != 51 && a[i]!= 52 && a[i]!=53
            && a[i] != 54 && a[i] != 55 && a[i]!=56
            && a[i] != 57) ← geht auch schöner:
        num= false; ✓
    }
    if (num == true)
        return 1; ← oder return true
    else {
        return 0;
        }
}
int main() { cout << IsNumber(a) << endl;
```

* char a[255]={48}
//oder so etwas

4