Aufgabe 3

a) Brute-Force-Algorithmus

```
2dMatchingBF(T,P):
    positions = []
    rightMatches = true
    for b=0 to T.length-P.length
        for a=0 to T.length-P.length
            for y=0 to P.length
                for x=0 to P.length
                    if P[y][x] != T[b+y][a+x]
                        rightMatches= false
                        break
                if !rightMatches
                    break
            if rightMatches
                positions.add((a,b))
            rightMatches = true
    return positions
```

Laufzeit:
Im schlimmsten Fall enthalten sowohl P als auch T immer nur den gleichen Buchstaben. In diesem Fall müssen für jede der n^2 Zellen in T m^2 Vergleiche durchgeführt werden, was zu einer Laufzeit von $O(m^2 * n^2)$ führt.

b) Rabin-Karp-Algorithmus

Modifizierung:

- Zur Umwandlung von P in eine Zahl stelle man sich das Feld von P als eine Zeichenkette mit aneinandergereihten Zeilen vor. Dann lässt sich eine Hashfunktion bzw. das Horner-Schema auf diesen String anwenden.
- Genauso geht man bei den Teilwörtern der Größe $m \times m$ in T vor, wobei alle Nachfolger von t_0 mit der Formel $t_{j+1} = d * t_j - a_j * d^m + a_{j+m}$ berechnet werden können.
- Der Rest des Algorithmus kann analog zur eindimensionalen Variante verwendet werden.

Laufzeit: $m = |P|$ $n = |T|$

- Umwandlung von P in Zahl: $O(m^2)$
- Umwandlung von t_0 in Zahl: $O(m^2)$
- Umwandlung aller zu P gleich großen Teilfelder von T: $O(n^2)$
- $(n - m + 1)^2$ Vergleiche: $O(n^2)$
- für jede Übereinstimmung: $O(m^2)$
- insgesamt: $O(n^2 * m^2)$