Aufgabe 1

swap1 – Die Werte sind außerhalb nicht vertauscht.
Begründung: Die Werte a und b werden *by value* übergeben, sodass der Aufrufer nur die Werte, aber nicht deren Speicheradresse übergibt. Entsprechend werden nur die Speicheradressen geändert, die a und b innerhalb der Methode verwenden.

swap2 – Die Werte werden vertauscht.
Begründung: Arrays werden *by reference* übergeben. Eine neuzuweisung einzelner Elemente im Array führt daher auch außerhalb der Methode zur Wert-Änderung.

swap3 – Die Werte werden nicht getauscht.
Begründung: Die Variable *arr* wird neu zugewiesen, wodurch die Referenz des übergebenen Arrays verloren geht und sich das alte Array außerhalb der Methode auf der Adresse bestehen bleibt.

swap4 – Die Werte werden getauscht.
Begründung: Die Werte werden wie bei swap2 nur einzeln neu zugewiesen, die Array-Instanz bleibt jedoch die selbe.
Die Zuweisung der Elemente mit den + und – Operationen sorgt für eine Vertauschung der Werte. Dies geht jedoch nur mit Zahlen-Typen.

Aufgabe 2a

```
m(2, 25) = m(25, 2) + m(2, 25 – 2) = 50 + 288 = 338
m(25, 2) = 25 * 2 = 50
m(2, 23) = m(23, 2) + m(2, 23 – 2) = 46 + 242 = 288
m(23, 2) = 23 * 2 = 46
m(2, 21) = m(21, 2) + m(2, 21 – 2) = 42 + 200 = 242
m(21, 2) = 21 * 2 = 42
m(2, 21) = m(19, 2) + m(2, 19 – 2) = 38 + 162 = 200
m(19, 2) = 19 * 2 = 38
m(2, 21) = m(17, 2) + m(2, 17 – 2) = 34 + 128 = 162
m(17, 2) = 17 * 2 = 34
m(2, 21) = m(15, 2) + m(2, 15 – 2) = 30 + 98  = 128
m(15, 2) = 15 * 2 = 30
m(2, 21) = m(13, 2) + m(2, 13 – 2) = 26 + 72  =  98
m(13, 2) = 13 * 2 = 26
m(2, 21) = m(11, 2) + m(2, 11 – 2) = 22 + 50  =  72
m(11, 2) = 11 * 2 = 22
m(2, 21) = m( 9, 2) + m(2,  9 – 2) = 18 + 32  =  50
m( 9, 2) =  9 * 2 = 18
m(2, 21) = m( 7, 2) + m(2,  7 – 2) = 14 + 18  =  32
m( 7, 2) =  7 * 2 = 14
m(2, 21) = m( 5, 2) + m(2,  5 – 2) = 10 +  8  =  18
m( 5, 2) =  5 * 2 = 10
m(2, 21) = m( 3, 2) + m(2,  3 – 2) =  6 +  2  =   8
m( 3, 2) =  3 * 2 = 6
m( 2, 1) =  2 * 1 = 2
```

Aufgabe 2b

[x] Die Methode m ist endrekursiv
[x] Der Aufruf der Methode m kann (abhängig von den übergebenen Werten) zum Überlauf des Stack-Speichers führen.
[ ] Der Aufruf der Methode m mit negativen Werten für a und b führt grundsätzlich zu einem Überlauf des Stack-Speichers.
[x] Wenn a und b initial die gleichen Werte haben, dann ist der Rückgabewert eine Quadratzahl.

Aufgabe 3

```
static int digitSum(int n) {
    int result = 0;
    while (n > 0) {
        result += n % 10;
        n = n / 10;
    }
    return result;
}
```

Aufgabe 5c

[ ] Wenn eine Lösung existiert, findet der Algorithmus immer eine Lösung.
[x] Wenn mehrere Wege zum Ziel führen und der Algorithmus einen Weg durch das Labyrinth findet, dann hat dieser minimale Länge.