Aufgabe 4

(a) Betrachten Sie das folgende Code-Beispiel (in Java-Notation):

```java
int mystery(int n) {
  int a = 0, b = 0;
  int i = 0;
  while (i < n) {
    a = b + i;
    b = a;
    i = i + 1;
  }
  return a;
}
```

Bestimmen Sie die asymptotische worst-case Laufzeit des Code-Beispiels in $\mathcal{O}$-Notation bezüglich der Problemgröße n. Begründen Sie Ihre Antwort.

> Die asymptotische worst-case Laufzeit des Code-Beispiels in $\mathcal{O}$-Notation ist $\mathcal{O}(n)$.
>
> Die `while`-Schleife wird genau n mal ausgeführt. In der Schleife wird die Variable `i` in der Zeile `i = i + 1;` inkrementiert. `i` wird mit 0 initialisiert. Die `while`-Schleife endet wenn `i` gleich groß ist wir `n`.

(b) Betrachten Sie das folgende Code-Beispiel (in Java-Notation):

```java
int mystery(int n) {
  int r = 0;
  while (n > 0) {
    int y = n;
    int x = n;
    for (int i = 0; i < y; i++) {
      for (int j = 0; j < i; j++) {
        r = r + 1;
      }
      r = r - 1;
    }
    n = n - 1;
  }
  return r;
```

Bestimmen Sie für das Code-Beispiel die asymptotische worst-case Laufzeit in $\mathcal{O}$-Notation bezüglich der Problemgröße n. Begründen Sie Ihre Antwort.

> `while`: n-mal
>
> **1.** `for`: $n, n-1, \ldots, 2, 1$
>
> **2.** `for`: $1, 2, \ldots, n-1, n$
>
> $n \times n \times n = \mathcal{O}(n^3)$

(c) Bestimmen Sie eine asymptotische Lösung (in Θ-Schreibweise) für die folgende Rekursionsgleichung:

$$T(n) = T\left(\frac{n}{2}\right) + \frac{1}{2}n^2 + n$$

$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$

a**:** $a = 1$
b**:** $b = 2$
$f(n)$**:** $f(n) = \frac{1}{2}n^2 + n$

$\log_b a = \log_2 1 = 0$

Erster Fall: $f(n) \in \mathcal{O}\left(n^{\log_b a - \varepsilon}\right)$

$\frac{1}{2}n^2 + n \notin \mathcal{O}\left(n^{-1}\right)$

Zweiter Fall: $f(n) \in \Theta\left(n^{\log_b a}\right)$

$\frac{1}{2}n^2 + n \notin \Theta(1)$

Dritter Fall: $f(n) \in \Omega\left(n^{\log_b a + \varepsilon}\right)$

$\varepsilon = 2$
$\frac{1}{2}n^2 + n \in \Omega\left(n^2\right)$