Aufgabe 3

Die folgende Seite enthält Software-Quellcode, der einen Algorithmus zur binären Suche implementiert. Dieser ist durch Inspektion zu überprüfen. Im Folgenden sind die Regeln der Inspektion angegeben.

RM1	(Dokumentation)	Jede Quellcode-Datei beginnt mit einem Kommentar, der den Klassennamen, Versionsinformationen, Datum und Urheberrechtsangaben enthält.
RM2	(Dokumentation)	Jede Methode wird kommentiert. Der Kommentar enthält eine vollständige Beschreibung der Signatur so wie eine Design-by-Contract-Spezifikation.
RM3	(Dokumentation)	Deklarationen von Variablen werden kommentiert.
RM4	(Dokumentation)	Jede Kontrollstruktur wird kommentiert.
RM5	(Formatierung)	Zwischen einem Schlüsselwort und einer Klammer steht ein Leerzeichen.
RM6	(Formatierung)	Zwischen binären Operatoren und den Operanden stehen Leerzeichen.
RM7	(Programmierung)	Variablen werden in der Anweisung initialisiert, in der sie auch deklariert werden.
RM8	(Bezeichner)	Klassennamen werden groß geschrieben, Variablennamen klein.

```
/**
 * BinarySearch.java
 *
 * Eine Implementierung der "Binaere Suche"
 * mit einem iterativen Algorithmus
 */
class BinarySearch {

  /**
   * BinaereSuche
   * a: Eingabefeld
   * item: zusuchendesElement
   * returnValue: der Index des zu suchenden Elements oder -1
   *
   * Vorbedingung:
   * a.length > 0
   * a ist ein linear geordnetes Feld:
   * For all k: (1 <= k < a.length) ==> (a[k-1] <=a [k])
   *
   * Nachbedingung:
   * Wenn item in a, dann gibt es ein k mit a[k] == item und returnValue == k
   * Genau dann wenn returnValue == -1 gibt es kein k mit 0 <= k < a.length
   * und a[k]==item.
   */
```

```
public static int binarySearch(float a[], float item) {

  int End;  // exklusiver Index fuer das Ende des
            // zudurchsuchenden Teils des Arrays
  int start = 1; // inklusiver Index fuer den Anfang der Suche
  End = a.length;

  // Die Schleife wird verlassen, wenn keine der beiden Haelften das
  // Element enthaelt.
  while(start < End) {

    // Teilung des Arrays in zwei Haelften
    // untere Haelfte: [0,mid[
    // obere Haelfte: ]mid,End[
    int mid = (start + End) / 2;

    if (item > a[mid]) {
      // Ausschluss der oberen Haelfte
      start = mid + 1;
    } else if(item < a[mid]) {
      // Ausschluss der unteren Haelfte
      End = mid-1;
    } else {
      // Das gesuchte Element wird zurueckgegeben
      return (mid);
    }
  } // end of while

  // Bei Misserfolg der Suche wird -1 zurueckgegeben
  return (-1);
 }
}
```

(a) Überprüfen Sie durch Inspektion, ob die obigen Regeln für den Quellcode eingehalten wurden. Erstellen Sie eine Liste mit allen Verletzungen der Regeln. Geben Sie für jede Verletzung einer Regel die Zeilennummer, Regelnummer und Kommentar an, z. B. (07, RM4, while nicht kommentiert). Schreiben Sie nicht in den Quellcode.

Zeile	Regel	Kommentar
3-8	RM1	Fehlen von Versionsinformationen, Datum und Urheberrechtsangaben
11-26	RM2	Fehlen der Invariante in der Design-by-Contract-Spezifikation
36,46	RM5	Fehlen des Leerzeichens vor der Klammer
48	RM6	Um einen binären (zweistellige) Operator handelt es sich im Code-Beispiel um den Subtraktionsoperator: `mid-1` . Hier fehlen die geforderten Leerzeichen.
32	RM7	Die Variable `End` wird in Zeile 32 deklariert, aber erst in Zeile initialisiert `End = a.length;`
32	RM8	Die Variable `End` muss klein geschrieben werden.

(b) Entspricht die Methode `binarySearch` ihrer Spezifikation, die durch Vor-und Nachbedingungen angeben ist? Geben Sie gegebenenfalls Korrekturen der Methode an.

Korrektur der Vorbedingung

Die Vorbedingung ist nicht erfüllt, da weder die Länge des Feldes `a` noch die Reihenfolge der Feldeinträge geprüft wurden.

```
if (a.length <= 0) {
  return -1;
}

for (int i = 0; i < a.length; i ++) {
  if ( a[i] > a[i + 1]) {
    return -1;
  }
}
```

Korrektur der Nachbedingung

`int start` muss mit `0` initialisiert werden, da sonst `a[0]` vernachlässigt wird.

(c) Beschreiben alle Kommentare ab Zeile 24 die Semantik des Codes korrekt? Geben Sie zu jedem falschen Kommentar einen korrigierten Kommentar mit Zeilennummer an.

Zeile	Kommentar im Code	Korrektur
34-35	`// Die Schleife wird verlassen, wenn keine der beiden Haelften das Element enthaelt.`	`// Die Schleife wird verlassen, wenn keine der beiden Haelften das Element enthaelt oder das Element gefunden wurde.`
44	`// Ausschluss der oberen Haelfte`	`// Ausschluss der unteren Haelfte`
47	`// Ausschluss der unteren Haelfte`	`// Ausschluss der oberen Haelfte`
50	`// Das gesuchte Element wird zurueckgegeben`	`// Der Index des gesuchten Elements wird zurueckgegeben`

(d) Geben Sie den Kontrollflussgraphen für die Methode `binarySearch` an.

(e) Geben Sie maximal drei Testfälle für die Methode `binarySearch` an, die insgesamt eine vollständige Anweisungsüberdeckung leisten.

Die gegebene Methode: `binarySearch(a[], item)`

Testfall

(i) Testfall: `a[] = {1, 2, 3}, item = 4`

(ii) Testfall: `a[] = {1, 2, 3}, item = 2`