Aufgabe 1: Funktionssignaturen (10 Punkte)

Betrachten Sie das folgende Programmfragment:

```
1   x1 := Foo1(42)
2   Foo2(x1)
3   s1 := []string{"Hallo"}
4   s2 := []string{"Welt"}
5   if Foo3(s1) {
6     x1 = Foo4(Foo3(s2), 15)
7   }
8   e1, e2 := Foo5(s1[0])
9   fmt.Printf("%d\n", x1+e1+e2)
```

Welche Signaturen haben die Funktionen `Foo1` bis `Foo5`?
Anmerkung: Die *Signatur* einer Funktion ist die erste Zeile, in der die Argument- und Rückgabetypen definiert werden. Hier ist also gefragt, welche Typen die Funktionen erwarten und liefern. Sie können davon ausgehen, dass Funktionen, deren Ergebnis nicht verwendet wird, auch keinen Rückgabetyp haben.

Lösung

Beobachtungen und Ergebnisse:

- `x1`, `e1` und `e2` müssen `int` sein, da sie am Ende mit `%d` ausgegeben werden.
- Deswegen gilt: Signatur von `Foo1`: `func Foo1(int) int`.
- Die Signatur von `Foo2` ist `func Foo2(int)`, weil `x1` vom Typ `int` ist.
- `s1` ist eine String-Slice und `Foo3` wird in einem `if` verwendet. Daher muss die Signatur von `Foo3` lauten: `Foo3([]string) bool`.
- `Foo4` ist `Foo4(bool, int) int`, da es `Foo3` und eine Zahl erwartet und eine Zahl nach `x1` schreibt.
- `Foo5` erwartet mit `s1[0]` einen String und liefert mit `e1` und `e2` zwei Zahlen, also `Foo5(string) (int, int)`.

Aufgabe 2: Fehlersuche: Compilerfehler (10 Punkte)

Das folgende Progamm enthält eine Reihe an Fehlern, durch die es nicht compiliert. Markieren Sie alle Zeilen, die einen Fehler enthalten und erläutern Sie kurz, was jeweils falsch ist.

```
1  package fehlersuche1
2
3  import "fmt"
4
5  func Foo(x int) {
6    return x + 3
7  }
8
9  func Bar(x, y int) string {
10   return fmt.Sprintf("%d␣+␣%d␣==␣%d", x, y, foo(x+y))
11 }
12
13 func FooBar()
14   s = "Zahlen"
15   x := s[0]
16   y := s[3]
17   fmt.Println(Bar(x, int(y)))
18 }
```

Hinweis: Es geht hier nicht um inhaltliche Fehler, nur um Syntaxfehler.
Anmerkung: Für jede falsch markierte Zeile gibt es Punktabzug!

Lösung

Im Programm sind folgende Fehler:

- Zeile 5: Der Rückgabetyp von `Foo` fehlt.
- Zeile 10: `Foo` ist falsch geschrieben.
- Zeile 13: Geschweifte Klammer fehlt.
- Zeile 14: Die Variable muss mit `:=` definiert werden.
- Zeile 15: Typumwandlung fehlt (hier oder in Zeile 17).

Aufgabe 3: Fehlersuche: Inhaltliche Fehler (10 Punkte)

Die folgende Funktion ist zwar syntaktisch korrekt, sie erfüllt aber nicht ihre Aufgabe. Erläutern Sie den/die Fehler und machen Sie einen Vorschlag zur Korrektur.

```
1  // Contains liefert true, falls die Liste die Zahl x
       genau n mal enthaelt.
2  func Contains(list []int, x, n int) bool {
3    counter := 0
4    for el := range list {
5      if el == x {
6        el = counter + 1
7        if counter == n {
8          return true
9        }
10     }
11   }
12   return false
13 }
```

Anmerkung: Ihr Korrekturvorschlag muss kein syntaktisch korrekter Code sein. Eine Erklärung in Worten genügt.

Lösung

Es gibt drei Fehler in diesem Code:

1. Die Schleife untersucht die Positionen und nicht die Elemnte. Es müsste `for _,el := range list` heißen.
2. Die Anweisung `el = counter + 1` hat keinen Effekt. Stattdessen müsste der Zähler erhöht werden (`counter++`).
3. Der Test `if counter == n` kommt zu früh. Die Funktion bricht dann ab, obwohl noch weitere `x` kommen könnten. Richtig wäre ein `return counter == n` am Ende.

Eine korrekte Version der Funktion wäre z.B. die folgende:

```
1  // Contains liefert true, falls die Liste die Zahl x
       genau n mal enthaelt.
2  func Contains(list []int, x, n int) bool {
3    counter := 0
4    for _, el := range list {
5      if el == x {
6        counter++
7      }
8    }
9    return counter == n
10 }
```

Aufgabe 4: Programmverständnis (10 Punkte)

Erläutern Sie, was die folgende Funktion berechnet. Geben Sie eine möglichst allgemeine bzw. abstrakte Erklärung an. Erklären Sie auch, mit welcher Art von Argumenten diese Funktion sinnvoll arbeitet.

```
1  func Foo(m, n int) int {
2    if m < n {
3      return 0
4    }
5    return 1 + Foo(m-n, n)
6  }
```

Lösung

Die Funktion berechnet die Division m geteilt durch n für $n \geq 0$ und $m > 0$.

Aufgabe 5: Datenstrukturen (5 Punkte)

Entwerfen Sie Datenstrukturen, mit denen möglichst klar eine Datenbank für Studierende, Kurse und Vorlesungen gebaut werden kann.
Für Studierende soll dabei ein Name und eine Liste von Vorlesungen gespeichert werden, für Vorlesungen der Name und die damit assoziierten Credits und für Kurse der Name und die teilnehmenden Studierenden.

Lösung

Eine mögliche Lösung wäre die folgende Sammlung von Structs:

```
type Student struct {
  Name     string
  Lectures []Lecture
}

type Course struct {
  Name     string
  Students []Student
}

type Lecture struct {
  Name    string
  Credits int
}
```