Aufgabe 1: Grundlagen (5 Punkte)

Welche der folgenden Behauptungen über die Programmiersprache Go sind wahr, welche falsch?

Behauptung	**wahr**	**falsch**
`int16` ist ein Datentyp in Go.	**X**	
`float64` ist ein Datentyp in Go.	**X**	
Eine Funktionssignatur sagt nichts über den Rückgabetyp der Funktion aus.		**X**
Bei der Definition von Variablen muss der Datentyp immer feststehen.	**X**	
`range`-Schleifen haben keinen Schleifenzähler.		**X**

Anmerkung: Korrekt angekreuzte Zeilen geben einen Punkt, für falsch angekreuzte Zeilen wird ein Punkt abgezogen.

Aufgabe 2: Signaturen (10 Punkte)

Betrachten Sie das folgende Programmfragment:

```
1   x1 := Foo1("Hallo", 15)
2   x2 := Foo2(x1)
3   x3 := Foo3(127, x2)
4   if x2 {
5     x3 = append(x3, Foo1("Welt", x1))
6   }
7   x1 += Foo4(x2, true)
8   Foo5(x2 && x1 != Foo4(x2, x2))
9   return x2 && !(x1 > x3[0])
```

Welche Signaturen haben die Funktionen `Foo1` bis `Foo5`? Welchen Rückgabetyp liefert das `return`?

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:

- `func Foo1(string, int) int`:
  - Zeile 1: `Foo1` erwartet `string` und `int`. Die Funktion liefert `x1`.
  - Zeile 5: `x1` wird wieder an `Foo1` übergeben, muss also `int` sein.
- `func Foo2(int) bool`:
  - Zeile 2: `Foo2` erwartet `int`, weil `x1` vom Typ `int` ist.
  - Zeile 4: `Foo2` liefert `bool`, weil `x2` in einem `if` verwendet wird.
- `func Foo3(int, bool) []int`:
  - Zeile 3: `Foo3` erwartet `int` und `bool`, da `x2` vom Typ `bool` ist.
  - Zeile 5: `Foo3` liefert `[]int`, weil `x1` vom Typ `int` ist und mittels `append` an `x3` angehängt wird.
- `Foo4(bool, bool) int`:
  - Zeile 7: `Foo4` erwartet `bool, bool`, weil `x2` vom Typ `bool` ist.
  - Zeile 7: `Foo2` liefert `int`, weil `x1` vom Typ `int` ist.
- `Foo5(bool)`:
  - Zeile 8: `Foo5` erwartet `bool`, weil der Ausdruck im Aufruf ein `&&` ist.
  - Zeile 8: `Foo5` liefert nichts.
- Die Funktion liefert `bool`, weil der Ausdruck im `return` ein `&&` ist.

Aufgabe 3: Fehlersuche: Compilerfehler (10 Punkte)

Der folgende Code enthält eine Reihe an Fehlern, durch die er 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) int {
6    return x := 3
7  }
8
9  func Bar(x, y int) string {
10   return string(Foo(5)))
11 }
12
13 Func FooBar() {
14   s := 'Huhu'
15   for x := range([]int{1,2,3,4,5}) {
16     fmt.Println(x)
17   }
18   s += y
19 }
```

Hinweis: Es geht hier nur um Syntaxfehler. Für jede falsch markierte Zeile gibt es Punktabzug!

Lösung

Hier ist eine Version, in der die Fehler markiert und korrigiert sind.

```
1  package fehlersuche1
2
3  import "fmt"
4
5  func Foo(x int) int {
6    return 3 // := während return geht nicht.
7  }
8
9  func Bar(x, y int) string {
10   return string(Foo(5)) // Klammer zu viel.
11 }
12
13 func FooBar() { // func groß geschrieben.
14   s := "Huhu" // einzelne statt doppelter Anführungszeichen
15   for x := range []int{1, 2, 3, 4, 5} {
16     fmt.Println(x)
17   }
18   s += "" // y ist nicht definiert.
19 }
```

Aufgabe 4: Fehlersuche: Inhaltliche Fehler (5 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  // Sorted liefert true, falls die Liste aufsteigend sortiert ist.
2  func Sorted(list []int) bool {
3    for _, el := range list[1:] {
4      if el < list[el] {
5        return false
6      }
7    }
8    return true
9  }
```

Anmerkung: Ihre Korrektur muss nicht syntaktisch korrekt sein. Eine Erklärung in Worten genügt.

Lösung

Eine korrekte Version der Funktion wäre z.B. die Folgende. Hier sind auch die Fehler markiert:

```
1  // Sorted liefert true, falls die Liste aufsteigend sortiert ist.
2  func Sorted(list []int) bool {
3    // Sonderfall für leere Liste hat gefehlt:
4    if len(list) == 0 {
5      return true
6    }
7    // Es wurde nur el verwendet, das enthält aber
8    // keine Position, sondern einen Wert.
9    for i, el := range list[1:] {
10     if el < list[i] {
11       return false
12     }
13   }
14   return true
15 }
```

Aufgabe 5: Programmverständnis (5 Punkte)

Erläutern Sie, was die Funktion `Foo` im folgenden Programmfragment berechnet. Geben Sie eine möglichst allgemeine bzw. abstrakte Erklärung an.

```
1  func Bar(n, i int) int {
2    if n < 0 || i == 0 {
3      return -1
4    }
5    if i*i == n {
6      return i
7    }
8    return Bar(n, i-1)
9  }
10
11 func Foo(n int) int {
12   return Bar(n, n)
13 }
```

Lösung

Falls n eine Quadratzahl ist, liefert `Foo` die Quadratwurzel aus n, ansonsten -1.

Aufgabe 6: Rekursion (10 Punkte)

Betrachten Sie die folgende Funktion:

```
1 func Foo(n, c int) int {
2   if n == 0 {
3     return 0
4   }
5   return Foo(n/10, c+1) + c*(n%10)
6 }
```

Beschreiben Sie in Worten, was die Funktion berechnet.
Berechnen Sie außerdem beispielhaft die Werte der Funktion für $n = 10$, $n = 11$ und $n = 201$ mit $c = 1$. Geben Sie dabei die Zwischenergebnisse der rekursiven Aufrufe mit an.

Lösung

Die Funktion berechnet die gewichtete Quersumme der Zahl n. D.h. es wird die Summe der Ziffern von n berechnet, wobei die Ziffern von rechts nach links mit $c, c+1, \ldots$ gewichtet werden.

Beispielrechnungen:

$$\begin{array}{lcl} Foo(0,3) & = & 0 \\ Foo(1,2) & = & Foo(0,3) + 2 \cdot 1 = 2 \\ Foo(10,1) & = & Foo(1,2) + 1 \cdot 0 = 2 \end{array}$$

$$\begin{array}{lcl} Foo(0,3) & = & 0 \\ Foo(1,2) & = & Foo(0,3) + 2 \cdot 1 = 2 \\ Foo(11,1) & = & Foo(1,2) + 1 \cdot 1 = 3 \end{array}$$

$$\begin{array}{lcl} Foo(0,4) & = & 0 \\ Foo(2,3) & = & Foo(0,4) + 3 \cdot 2 = 6 \\ Foo(20,2) & = & Foo(0,3) + 2 \cdot 0 = 6 \\ Foo(201,1) & = & Foo(1,2) + 1 \cdot 1 = 7 \end{array}$$