

Aufgabe 1: Grundlagen**(5 Punkte)**

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

Behauptung	wahr	falsch
float ist ein Datentyp in Go.		
Ein string kann mit append verlängert werden.		
Funktionsparameter müssen immer verwendet werden.		
Variablen können oft ohne explizite Typangabe deklariert werden.		
Funktionen haben immer einen Namen		

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  Foo1(Foo2(42))
2  x1 := Foo3(true, 23)
3  if len(x1) > 0 {
4      fmt.Println(Foo2(75 - Foo4()))
5  }
6  return x1 == "" && Foo1(len(x1))
```

Welche Signaturen haben die Funktionen Foo1 bis Foo4? 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.

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 Never {
6     fmt.Println("Never")
7 }
8
9 func Count(n int) string {
10    if n = 0 {
11        return ""
12    }
13    return Count(n-1) += fmt.Sprint(n)
14 }
15
16 func Upwards(s int) {
17     println(s + Count(5))
18 }
```

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

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 // AllEqual liefert true, falls alle Zeichen in `s` gleich sind.
2 func AllEqual(s []int) bool {
3     for el := range s {
4         if el != s[0] {
5             return false
6         }
7     }
8
9     return len(s) != 0
10 }
```

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

Aufgabe 5: Programmverständnis**(10 Punkte)**

Erläutern Sie die Funktionsweise der folgenden Datenstruktur. Geben Sie eine Erklärung für die Struktur und erläutern Sie kurz den Zweck der gegebenen Methoden.

```
1 type S struct {
2     els []string
3 }
4
5 func (s S) C(el string) bool {
6     for _, e := range s.els {
7         if e == el {
8             return true
9         }
10    }
11    return false
12 }
13
14 func (s *S) A(el string) {
15     if !s.C(el) {
16         s.els = append(s.els, el)
17     }
18 }
19
20 func (s S) U(o S) S {
21     r := S{}
22     for _, e := range s.els {
23         r.A(e)
24     }
25     for _, e := range o.els {
26         r.A(e)
27     }
28     return r
29 }
30
31 func (s S) I(o S) S {
32     r := S{}
33     for _, e := range s.els {
34         if o.C(e) {
35             r.A(e)
36         }
37     }
38     return r
39 }
```

Aufgabe 6: Rekursion**(10 Punkte)**

Betrachten Sie die folgende Funktion:

```
1 func Foo(s string, c byte) int {
2     if len(s) == 0 {
3         return 0
4     }
5
6     count := 0
7     if s[0] == c {
8         count = len(s)
9     }
10
11    return count + Foo(s[1:], c)
12 }
```

Beschreiben Sie in Worten, was die Funktion berechnet.

Hinweis: Der Typ `byte` repräsentiert ein einzelnes Zeichen. Die Funktion erwartet also einen String und ein Zeichen als Eingabe. Mögliche Beispielrechnungen: `Foo("a", 'a')`, `Foo("ab", 'a')`, `Foo("aa", 'a')`, `Foo("aba", 'a')`.