Aufgabe 1: Sortierverfahren

Sortieren Sie die folgende Liste von Zahlen aufsteigend mit dem Verfahren *Bubble Sort*:

$$17, 22, 13, 38, 35, 12$$

Geben Sie die Liste nach jedem Durchlauf der inneren Schleife an.

Aufgabe 2: Funktionssignaturen

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.

func Foo1(int) int

func Foo2(int)

func Foo3(string) bool

func Foo4(bool, int)

func Foo5(string) (int, int)

Aufgabe 3: Fehlersuche: Compilerfehler

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!

Aufgabe 4: Fehlersuche: Inhaltliche Fehler

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.

Aufgabe 5: Programmverständnis

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 }
```

Division m : n m≥0 n>0

Aufgabe 6: Datenstrukturen

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.

```
type Student struct {
    Name        string
    Vorlesungen []Vorlesungen }

type Vorlesungen struct{
    Name        string
    Credits     int

type Kurse struct {
    Name        string
    Students    []student
```