PRO1UEG*x* **Übungen zu Einführung in die Programmierung, Algorithmen und Datenstrukturen** **WS 25/26, Übung 5**

Abgabetermin: Mi, 26.11.2025

☐ **Gr. 1,** Dr. D. Auer **Name** _____ **Aufwand in h** _____
☐ **Gr. 2,** Dr. G. Kronberger
☐ **Gr. 3,** Dr. S. Wagner **Punkte** _____ **Kurzzeichen Tutor*in / Übungsleiter*in** ____ / ____

1. Feldverarbeitung mit offenen Feldparametern: Schnittmengen (8 Punkte)

Gegeben sind zwei beliebig große Felder `a1` und `a2`, die `n1` bzw. `n2` ganze Zahlen in aufsteigend sortierter Reihenfolge enthalten. Gesucht ist eine Pascal-Prozedur

```
PROCEDURE Intersect(a1: ARRAY OF INTEGER; n1: INTEGER;
                    a2: ARRAY OF INTEGER; n2: INTEGER;
                    VAR a3: ARRAY OF INTEGER; VAR n3: INTEGER);
```

die das Feld `a3` so befüllt, dass darin alle `n3` Zahlen in aufsteigend sortierter Reihenfolge enthalten sind, die sowohl in `a1` als auch in `a2` vorkommen. Im Fehlerfall (Überlauf in `a3`, Felder `a1` und `a2` sind nicht aufsteigend sortiert) soll `n3` auf –1 gestellt werden.

Beispiel:

`a1` =	1	2	3	5	8	13	…	`n1` = 6
`a2` =	1	3	5	7	9	…		`n2` = 5
`a3` =	1	3	5	…				`n3` = 3

Hinweis: Implementieren Sie eine Pascal-Funktion `IsSorted`, die prüft, ob die Werte in einem Feld `a` aufsteigend sortiert sind und verwenden Sie diese Funktion in der Prozedur `Intersect`.

2. Funktionen zur Zeichenkettenverarbeitung (1 + 2 + 4 Punkte)

Entwickeln Sie weitere Operationen zur Zeichenkettenbearbeitung (Datentyp `STRING`). Verwenden Sie dafür insbesondere die bereits vorhandenen Pascal-Standardfunktionen.

a) Implementieren Sie eine Funktion `WithoutLastChar`, die in der als Eingangsparameter `s` übergebenen Zeichenkette das letzte Zeichen entfernt und das Ergebnis als Funktionswert liefert.

b) Implementieren Sie eine Funktion `EqualsIgnoreCase`, die zwei Zeichenketten `a` und `b` auf Gleichheit prüft. Dabei soll die Groß/Kleinschreibung nicht berücksichtigt werden, d.h. für die Zeichenketten `a = 'Pascal'` und `b = 'PASCAL'` muss der Algorithmus `TRUE` liefern.

c) Implementieren Sie eine Funktion

```
FUNCTION CamelCase(words: ARRAY OF STRING; n: INTEGER): STRING;
```

die die `n` im String-Array `words` gespeicherten Wörter zu einer Zeichenkette in *camelCase* Schreibweise zusammensetzt und zurückliefert. Jedes Wort darf dabei nur aus Kleinbuchstaben (`a..z`), Großbuchstaben (`A..Z`) oder Ziffern (`0..9`) bestehen. Im Fehlerfall, z.B. wenn die übergebenen Wörter nicht korrekt sind oder die resultierende Zeichenkette zu lang ist, soll als Ergebnis die Zeichenkette `ERROR` zurückgeliefert werden.

Beispiele:

`words:`	`My`	`cAMEL`	`CaSe`	`name2`		➔ `myCamelCaseName2`
`n:`	4					

`words:`	`wrong`	`n#me`				➔ `ERROR`
`n:`	2					

3. Kaffeeautomat (7 + 2 Punkte)

Entwickeln Sie ein Steuerprogramm für einen Kaffeeautomaten. Der Automat nimmt 10- und 50-Cent-Münzen sowie 1-Euro-Münzen. Der Preis für einen Kaffee beträgt 40 Cent. Nach dem Einwurf des Geldes und dem Drücken der "Kaffee-Taste" wird ein Algorithmus gestartet. Beim Einwurf von weniger als 40 Cent soll der Algorithmus das Geld wieder auswerfen und eine entsprechende Fehlermeldung ausgeben. Ist der eingeworfene Betrag zu hoch, soll der Automat den Restbetrag zurückgeben. Er soll dabei möglichst wenige Münzen verwenden. Falls der Automat nicht über ausreichend Wechselgeld verfügt, soll er den eingeworfenen Betrag zurückgeben und eine entsprechende Fehlermeldung ausgeben. Falls dies dreimal hintereinander passiert, soll der Automat die Fehlermeldung "ERROR: Out of order!" ausgeben und auf keine weiteren Münzeinwürfe/Tastendrucke mehr reagieren.

Implementieren Sie einen Algorithmus mit Gedächtnis mit folgender Schnittstelle

```
PROCEDURE CoffeeButtonPressed(input: Coins; VAR change: Coins);
```

der das Drücken der "Kaffee-Taste" simuliert. Am Anfang soll der Automat über Wechselgeld in der Höhe von zehn 10-Cent- Münzen, fünf 50-Cent-Münzen und keine 1-Euro Münzen verfügen.

a) Implementieren Sie das Gedächtnis in Form von globalen Variablen.

b) Verpacken Sie Ihren Algorithmus mit Gedächtnis in eine Unit.

Hinweis:

Definieren Sie den Datentyp `Coins` entsprechend der Aufgabenstellung, so dass ein `Coins`-Datenobjekt die Anzahl der drei verschiedenen Münzen speichern kann.

Hinweise:

1. Geben Sie für alle Ihre Lösungen immer eine „Lösungsidee“ an.
2. Dokumentieren und kommentieren Sie Ihre Pascal-Programme.
3. Geben Sie immer auch Testfälle ab, an denen man erkennen kann, dass Ihr Pascal-Programm funktioniert, und dass es auch in Fehlersituationen entsprechend reagiert.