**Abgabetermin: Mi in der KW 46**

☐ **Gr. 1,** Dr. D. Auer
☐ **Gr. 2,** Dr. G. Kronberger **Name** _____ **Aufwand in h** _____
☐ **Gr. 3,** Dr. H. Gruber

**Punkte** _____ **Kurzzeichen Tutor / Übungsleiter** _____ / _____

## 1. Telefonverzeichnis (16 Punkte)

Implementieren Sie ein elektronisches Telefonverzeichnis, welches es ermöglicht, Einträge zu speichern, abzurufen und zu löschen. Verwenden Sie dazu die folgenden Deklarationen:

```
CONST
  max = 10;
TYPE
  Entry = RECORD
    firstName: STRING[20]
    lastName: STRING[30];
    phoneNumber: INTEGER;
  END; (*RECORD*)
  PhoneBook = ARRAY [1 .. max] OF Entry;
```

Das Telefonverzeichnis soll lückenlos gefüllt werden, auf eine Sortierung können Sie verzichten. Beim Löschen eines Eintrags ist die Lücke durch Verschieben der restlichen Einträge zu schließen. Zugriffe auf das Verzeichnis dürfen nur über die Prozeduren *AddEntry*, *DeleteEntry*, *SearchName*, *SearchNumber* und die Funktion *NrOfEntries* erfolgen (wählen Sie die Parameter passend):

- `PROC. AddEntry(...);`
  Erweitert das Verzeichnis um einen Eintrag. Bei einem Überlauf muss eine Fehlermeldung ausgegeben werden; das Verzeichnis darf in diesem Fall nicht verändert werden.
- `PROC. DeleteEntry(...);`
  Versucht, einen durch Vor- und Nachnamen gegebenen Eintrag zu entfernen. Ist dieser nicht vorhanden, so ist eine Fehlermeldung auszugeben.
- `PROC. SearchNumber(...);`
  Sucht einen Eintrag mit Vor- und Nachnamen. Ist kein solcher vorhanden, so ist eine Fehlermeldung auszugeben. Bei mehrfachen Einträgen sind alle auszugeben.
- `PROC. SearchName(...);`
  Sucht einen Eintrag nach der Telefonnummer. Ist kein Eintrag mit dieser Nummer vorhanden, so ist eine Fehlermeldung auszugeben.
- `FUNC. NrOfEntries(...): INTEGER;`
  Liefert die aktuelle Anzahl der Einträge.

## 2. Feldverarbeitung mit offenen Feldparametern (8 Punkte)

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

```
PROC. Merge(a1, a2: ARRAY OF INT.; VAR a3: ARRAY OF INT.; VAR n3: INT.);
```

die ein aufsteigend sortiertes Feld *a3* liefert, das alle *n3* Zahlen enthält, die in *a1* oder in *a2* aber nicht in *a1* und *a2* vorkommen (vgl. *XOR*). Beachten Sie, dass in jedem Feld (also auch im Ergebnisfeld *a3*) Werte mehrfach vorkommen können. Im Fehlerfall (Feld *a3* würde überlaufen) soll *n3* auf –1 gestellt werden. *Beispiel*:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *a1* = | 2 | 4 | 4 | 10 | 15 | 15 | |
| *a2* = | 3 | 4 | 5 | 10 | | | |
| *a3* = | 2 | 3 | 5 | 15 | 15 | | und *n3* = 5 |