PRG1x & ADE1x Üb. zu Einf. i d. Programmierung und Element. Alg. u. Datenstrukt. **WS 22/23, Übung 5**

Abgabetermin: Sa, 10.12.2022

☐ **Gr. 1,** DI (FH) G. Horn-Völlenkle, MSc **Name** _____ **Aufwand in h** _____

☐ **Gr. 2,** S. Schöberl, MSc

Punkte _____ **Tutor*in / Übungsleiter*in** ____ / ____

1. WWW-Zugriffszahlen (12 Punkte)

Ein Web-Server speichert bei jedem Zugriff auf eine Web-Seite die IP-Adresse, von der aus zugegriffen wurde. Zur Auswertung der Zugriffszahlen könnte man eine *einfach-verkettete* Liste auf Basis folgender Deklarationen verwenden:

```
CONST
  max = 4;
TYPE
  IPAddrNodePtr = ^IPAddrNode;
  IPAddrNode = RECORD
    next: IPAddrNodePtr;
    addr: ARRAY [1..max] OF BYTE;
    n: INTEGER; (* number of accesses from addr *)
  END; (* IPAddrNode *)
```

Entwickeln Sie ein Programm, welches die IP-Adressen einliest und diese mit ihren Häufigkeiten in Form einer nach IP-Adressen lexikographisch aufsteigend sortierten Liste speichert. Anschließend soll die Anzahl der IP-Adressen mit mehr als einem Zugriff ausgegeben werden. *Beispiel:* Speichern einer IP-Adresse in dem Feld `addr`:

IP-Adresse: „194.232.104.141“

Feld

addr:	194	232	104	141
	1	2	3	4

Hinweis: In der Datei IPAddr.zip sind Textdateien mit zeilenweise gespeicherten IP-Adressen enthalten. Lesen Sie den Inhalt einer Textdatei mittels Standardprozeduren *Read* und *ReadLn* und leiten Sie den Inhalt der Datei mit IP-Adressen auf die Standardeingabe um. *Beispiel*:

```
U:\>IPAddrCount.exe < ip2.txt
```

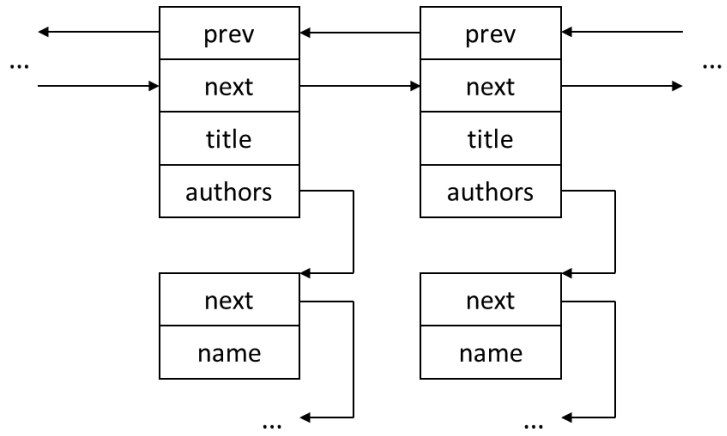
2. Bibliotheksverwaltung (Listen) (12 Punkte)

Im Rahmen einer Bibliotheksverwaltung soll ein Buch/Autoren-Verzeichnis erstellt werden. Das Verzeichnis soll eine Liste der Bücher und zu jedem Buch eine Liste seiner Autoren (Autorenverzeichnis) enthalten. Da weder die Anzahl der Bücher noch die Anzahl der Autoren je Buch bekannt ist, wählen Sie eine dynamische Datenstruktur (die beliebig wachsen und schrumpfen kann). Sie verwenden eine *doppelt-verketteten* Liste (mit Ankerelement und zyklischer Verkettung) für das Buchverzeichnis und eine *einfach-verkettete* Liste für das Autorenverzeichnis gemäß folgender Deklarationen:

```
TYPE
  BookNodePtr = ^BookNode;
  BookNode = RECORD
    prev, next: BookNodePtr;
    title: STRING;
    authors: AuthorNodePtr;
  END; (* BookNode *)
VAR
  bookIndex: BookNodePtr;
```

```
TYPE
  AuthorNodePtr = ^AuthorNode;
  AuthorNode = RECORD
    next: AuthorNodePtr;
    name: STRING;
  END; (* AuthorNode *)
```

Dadurch ergibt sich eine doppelt-verkettete Liste, bei der in jedem Knoten eine einfach-verkettete Liste mit mindestens einem Knoten ankert, gemäß folgender Abbildung:



Implementieren Sie diese Datenstruktur in Form eines Moduls mit (mindestens) folgenden Funktionen und Prozeduren:

(a) `PROCEDURE InsertBook(title: STRING; author: STRING);`

welche das Paar Autor/Buch in die Datenstruktur einfügt, wobei beide Listen bezgl. den Autornamen bzw. Buchtitel sortiert sein sollen. Hinweis: für ein Buch mit mehreren Autoren wird diese Prozedur mehrmals aufgerufen.

(b) `FUNCTION NrOfBooksOf(author: STRING): INTEGER;`

welche die Anzahl der Bücher eines Autors ermittelt.

(c) `PROCEDURE PrintAuthorsOf(title: STRING);`

welche die Autoren eines Buchs ausgibt.

(d) `PROCEDURE PrintAll;`

welche das gesamte Verzeichnis ausgibt.

(e) `PROCEDURE DisposeAll;`

welche den Speicher für alle Knoten der Datenstruktur freigibt.

Implementieren Sie ein Hauptprogramm, mit dem Sie die Funktionen und Prozeduren des Moduls testen.

Hinweise:

1. Geben Sie für alle Ihre Lösungen immer eine „Lösungsidee“ an.
2. Dokumentieren und kommentieren Sie Ihre Algorithmen.
3. Bei Programmen: Geben Sie immer auch Testfälle ab, an denen man erkennen kann, dass Ihr Programm funktioniert, und dass es auch in Fehlersituation entsprechend reagiert.