Abgabetermin: Mi, 10.12.2025

☐ **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. Einfach-verkettete Listen (4 Punkte)

Gegeben sind folgende Deklarationen für eine einfach-verkettete Liste von ganzen Zahlen.

```
TYPE
  ListNodePtr = ^ListNode;
  ListNode = RECORD
    next: ListNodePtr;
    data: INTEGER;
  END; (* ListNode *)
  ListPtr = ListNodePtr;
```

Gesucht ist eine Prozedur

```
PROCEDURE RemoveMax(VAR list: ListPtr; VAR maxNode: ListNodePtr);
```

die den Knoten mit dem größten Wert in der Komponente `data` aus der Liste `list` entfernt und diesen als Ausgangsparameter `maxNode` zurückgibt. Für eine leere Liste soll `maxNode` = `NIL` geliefert werden. Wenn mehrere Knoten den größten Wert enthalten, soll der erste dieser Knoten geliefert werden.

2. WWW-Zugriffszahlen (10 Punkte)

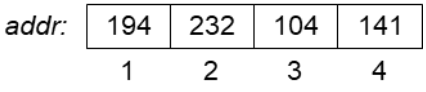
Ein Webserver speichert bei jedem Zugriff auf eine Webseite die IP-Adresse, von der aus zugegriffen wurde. Zur Auswertung der Zugriffszahlen kann man eine *einfach-verkettete Liste* auf Basis folgender Deklarationen verwenden:

```
CONST
  ip4max = 4;
TYPE
  IPAddr = ARRAY [1..ip4max] OF BYTE;
  IPAddrNodePtr = ^IPAddrNode;
  IPAddrNode = RECORD
    next: IPAddrNodePtr;
    addr: IPAddr;
    count: 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



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:

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

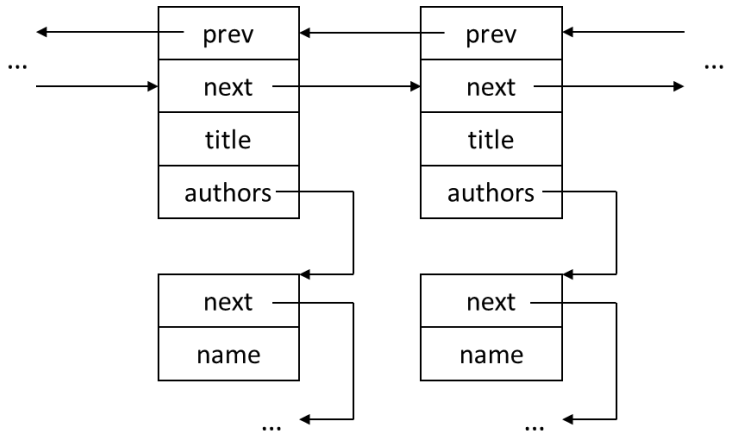
3. Bibliotheksverwaltung (Listen) (10 Punkte)

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

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

```
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 mit (mindestens) folgenden Funktionen und Prozeduren:

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

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

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

welche die Anzahl der Bücher eines Autors/einer Autorin ermittelt.

(c) `PROCEDURE PrintAll;`

welche das gesamte Verzeichnis ausgibt (alle Buchtitel mit allen Autor*innen).

(d) `PROCEDURE DisposeAll;`

welche den Speicher für alle Knoten der Datenstruktur freigibt.

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.