Programmierung **WS 2000/2001** **Prof. H. Lichter**
Th. von der Maßen

Übung 7
Musterlösung

Aufgabe 7.1

a), b), c)

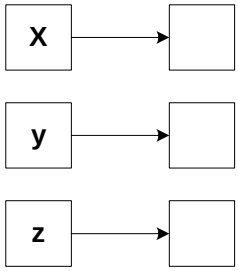
```
MODULE Mengenrelationen EXPORTS Main;

(* Dieses Programm realisiert Mengenrelationen.
   Autor           : Thomas von der Maßen, RWTH Aachen
   Umgebung        : PM-3, Windows 2000
   Erstellt        : 29.11.00  Letzte Aenderung: 30.11.00
*)


IMPORT SIO;

(* Deklariere Datentypen *)
TYPE Zeichen = ['a' .. 'z'];
     Buchstaben = SET OF Zeichen;

VAR menge1, menge2 := Buchstaben{};
    alle := Buchstaben{'a' .. 'z'};
    z : CHAR;
    ergebnis : BOOLEAN;


PROCEDURE Ausgabe(m: Buchstaben)=
(* Diese Prozedure gibt alle Elemente einer Menge aus *)
BEGIN
  FOR i:=FIRST(Zeichen) TO LAST(Zeichen) DO
    IF i IN m THEN
      SIO.PutChar(i);
      SIO.PutChar(' ');
    END;
  END;
END Ausgabe;


PROCEDURE Einlesen(VAR m: Buchstaben)=
(* Diese Prozedur liest solange Zeichen von der Tastatur ein und legt diese in
einer Menge ab,
bis '!' eingegeben wird. *)
BEGIN
  REPEAT
    z := SIO.GetChar();
    IF z IN alle THEN
      m := m + Buchstaben{z};
    END;
  UNTIL z = '!';
END Einlesen;


PROCEDURE DruckeMenu()=
(* Diese Prozedur zeigt das Benutzermenü auf dem Bildschirm an. *)
BEGIN
  SIO.PutText("+----------------------+"); SIO.Nl();
```

```
  SIO.PutText("| 1 - Gleichheit       |"); SIO.Nl();
  SIO.PutText("| 2 - Ungleichheit     |"); SIO.Nl();
  SIO.PutText("| 3 - Teilmenge        |"); SIO.Nl();
  SIO.PutText("| 4 - Echte Teilmenge  |"); SIO.Nl();
  SIO.PutText("| 5 - Obermenge        |"); SIO.Nl();
  SIO.PutText("| 6 - Echte Obermenge  |"); SIO.Nl();
  SIO.PutText("| 7 - Beenden          |"); SIO.Nl();
  SIO.PutText("+----------------------+"); SIO.Nl();
  SIO.PutText("Bitte treffen Sie eine Auswahl: ");
END DruckeMenu;

PROCEDURE Gleich(m1, m2: Buchstaben): BOOLEAN =
(* Prüft, ob die Mengen m1 und m2 Mengen gleich sind *)
BEGIN
  RETURN (Teilmenge(m1, m2) AND Teilmenge(m2, m1));
END Gleich;

PROCEDURE Ungleich(m1, m2: Buchstaben): BOOLEAN =
(* Prüft ob die Mengen m1 und m2 ungleich sind *)
BEGIN
  RETURN NOT(Gleich(m1, m2));
END Ungleich;

PROCEDURE Teilmenge(m1, m2: Buchstaben): BOOLEAN =
(* Prüft, ob die Menge m1 eine Teilmenge der Menge m2 ist *)
VAR istGleich: BOOLEAN;
BEGIN
  istGleich := TRUE;
  FOR i:=FIRST(Zeichen) TO LAST(Zeichen) DO
    (* Falls das Element i in m1 ist, dann muss es auch in m2 vorhanden sein *)
    IF i IN m1 THEN
      istGleich := istGleich AND (i IN m2);
    END;
  END;
  RETURN istGleich;
END Teilmenge;

PROCEDURE EchteTeilmenge(m1, m2: Buchstaben): BOOLEAN =
(* Prüft, ob die Menge m1 eine echte Teilmenge der Menge m2 ist *)
BEGIN
  RETURN (Teilmenge(m1, m2) AND Ungleich(m1, m2));
END EchteTeilmenge;

PROCEDURE Obermenge(m1, m2: Buchstaben): BOOLEAN =
(* Prüft, ob die Menge m1 eine Obermenge der Menge m2 ist *)
BEGIN
  RETURN Teilmenge(m2, m1);
END Obermenge;

PROCEDURE EchteObermenge(m1, m2: Buchstaben): BOOLEAN =
(* Prüft, ob die Menge m1 eine echte Obermenge der Menge m2 ist *)
BEGIN
  RETURN EchteTeilmenge(m2, m1);
END EchteObermenge;

BEGIN
  SIO.PutText("Geben Sie bitte die Elemente der 1. Menge ein (Abschliessen mit
'!'):");
  SIO.Nl();
  Einlesen(menge1);
  SIO.PutText("Geben Sie bitte die Elemente der 2. Menge ein (Abschliessen mit
'!'):");
  SIO.Nl();
  Einlesen(menge2);
```

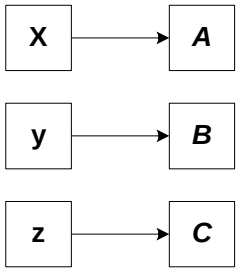
```
  SIO.PutText("Die Elemente der 1. Menge lauten: ");
  Ausgabe(menge1);
  SIO.Nl();
  SIO.PutText("Die Elemente der 2. Menge lauten: ");
  Ausgabe(menge2);
  SIO.Nl();

  ergebnis := TRUE;
  REPEAT
    DruckeMenu();
    (* Umgehung eines Bugs in Windows *)
    z := SIO.GetChar();
    z := SIO.GetChar();
    z := SIO.GetChar();
    CASE z OF
      '1' => ergebnis := Gleich(menge1, menge2); |
      '2' => ergebnis := Ungleich(menge1, menge2); |
      '3' => ergebnis := Teilmenge(menge1, menge2); |
      '4' => ergebnis := EchteTeilmenge(menge1, menge2); |
      '5' => ergebnis := Obermenge(menge1, menge2); |
      '6' => ergebnis := EchteObermenge(menge1, menge2); |
      '7' => SIO.PutText("Ende");
    ELSE
        SIO.PutText("Keine gültige Auswahl !!!");
        SIO.Nl();
    END;
    IF z # '7' THEN
      SIO.PutText("Ergebnis: "); SIO.PutBool(ergebnis); SIO.Nl();
    END;
  UNTIL z = '7';

END Mengenrelationen.
```

Aufgabe 7.2

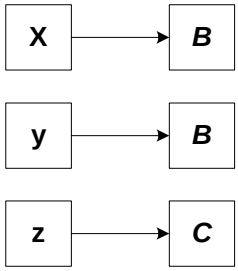
1



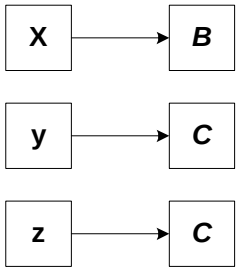
2



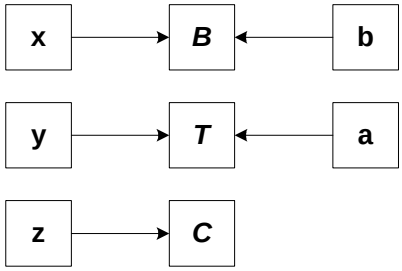
3



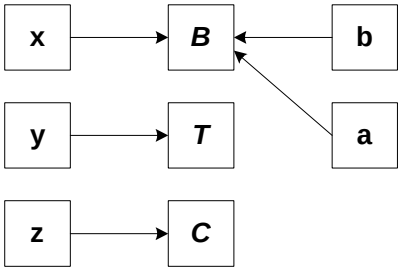
4



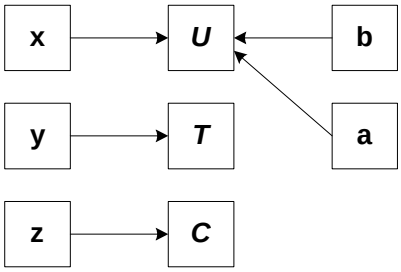
5



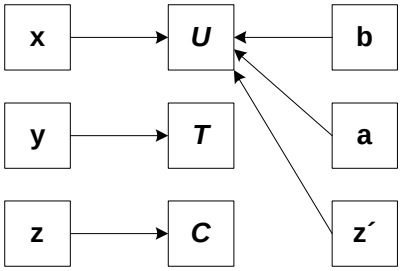
6



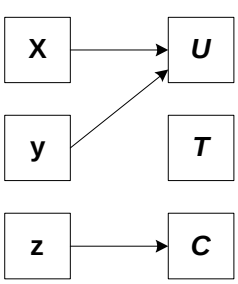
7



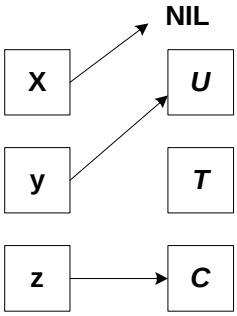
8



9



10



Ausgabe :

```
x = NIL
y = ´U´ ;
z = ´C´ :
```

`z := NEW(CharRef)`	Reservierung von Speicherplatz auf der Halde. Achtung: Der Inhalt z^ ist zwar zugreifbar, aber in einem nicht initalisierten Zustand
`z^ := ´C´`	Der Inhalt z^ ist nun auf das Zeichen ´C´ gesetzt
`x^ := y^`	Der Inhalt y^ wird nach x^ kopiert und ist somit zweimal im Speicher vorhanden, wenn x und y auf verschiedene Speicherbereiche weisen
`a := b`	Die Zeigervariablen verweist nun auf den selben Inhalt wie b.
`x := NIL`	NIL ist ein für jeden Zeigertyp definierter Adresswert und ist so interpretierbar, dass die Varibale x zwar in einem definierten Zustand, ihr aber kein Inhalt zugewiesen ist, also x inhaltsleer ist. Dieser Zustand kann mit `x = NIL` getestet werden.

Aufgabe 7.3

```
MODULE Boxrangliste EXPORTS Main;

(* Dieses Programm dient der Verwaltung einer Boxrangliste.
   Autor           : Moritz Schnizler, Thomas von der Maßen, RWTH Aachen
   Umgebung        : PM-3, Windows 2000
   Erstellt        : 29.11.00  Letzte Aenderung: 15.12.00
*)


IMPORT SIO;   (* Importiere Operationen fuer die Ein-/Ausgabe *)
IMPORT Text;  (* Importiere Operationen fuer Texte *)


TYPE Name = RECORD
              vorname : TEXT;
              nachname: TEXT;
            END;

     Boxer = RECORD
                name       : Name;
                nation     : TEXT;
                gewicht    : CARDINAL;
              END;

     RanglisteRef = REF RanglistenElement;

     RanglistenElement = RECORD
                           boxer    : Boxer;
                           naechster: RanglisteRef;
                         END;


VAR rangliste    : RanglisteRef; (* Anker der Club-Rangliste *)
    boxer,
    gewinner,
    verlierer    : Boxer;
    auswahl      : CHAR;
    dummy        : TEXT;
```

```
PROCEDURE InitialisiereRL(VAR liste: RanglisteRef)=
(* Initialisiert die Datenstruktur fuer eine Rangliste *)
BEGIN
  liste := NIL; (* Setze Anker der Rangliste auf NIL *)
END InitialisiereRL;


PROCEDURE SucheBoxerRL(liste: RanglisteRef; boxer: Boxer): RanglisteRef=
(* Sucht den gegebenen Boxer in der Rangliste liste und gibt einen Zeiger
   auf ihn zurueck. Der Zeiger hat den Wert NIL, wenn der Boxer nicht in der
   Liste vorhanden ist. *)

VAR resRLRef: RanglisteRef;
    gefunden: BOOLEAN;

BEGIN
   gefunden := FALSE;
   resRLRef := liste;
   WHILE (resRLRef # NIL) AND NOT gefunden DO

     (* Pruefe, ob der referenzierte Boxer der gesuchte ist *)
     IF (Text.Equal(resRLRef^.boxer.name.vorname, boxer.name.vorname)) AND
        (Text.Equal(resRLRef^.boxer.name.nachname, boxer.name.nachname)) AND
        (Text.Equal(resRLRef^.boxer.nation,  boxer.nation)) AND
        (resRLRef^.boxer.gewicht = boxer.gewicht)
     THEN
       gefunden := TRUE;
     ELSE
       resRLRef := resRLRef^.naechster;
     END;

   END; (* WHILE *)

   RETURN resRLRef;
END SucheBoxerRL;


PROCEDURE SucheVorgaengerRL(liste: RanglisteRef; boxer: Boxer): RanglisteRef=
(* Sucht den Vorgaenger des gegebenen Boxers in der Rangliste liste und gibt
   einen Zeiger auf ihn zurueck. Der Zeiger hat den Wert NIL, wenn der Boxer
keinen
   Vorgaenger hat, d.h. wenn er das erste Element oder nicht in der Liste ist.
*)

VAR vorRLRef, aktRLRef: RanglisteRef;
    gefunden          : BOOLEAN;

BEGIN
   gefunden := FALSE;
   vorRLRef := NIL;    (* Es gibt keinen Vorgaenger des ersten Elements! *)
   aktRLRef := liste;
   WHILE (aktRLRef # NIL) AND NOT gefunden DO

     (* Pruefe, ob der referenzierte Boxer der gesuchte ist *)
     IF (Text.Equal(aktRLRef^.boxer.name.vorname, boxer.name.vorname)) AND
        (Text.Equal(aktRLRef^.boxer.name.nachname, boxer.name.nachname)) AND
        (Text.Equal(aktRLRef^.boxer.nation, boxer.nation)) AND
        (aktRLRef^.boxer.gewicht = boxer.gewicht)
     THEN
       gefunden := TRUE;
     ELSE
       vorRLRef := aktRLRef;
       aktRLRef := aktRLRef^.naechster;
     END;
```

```
   END; (* WHILE *)

   (* Zeiger auf den Vorgaenger ist NIL, wenn Boxer nicht in der Liste ist *)
   IF NOT gefunden THEN vorRLRef := NIL; END;

   RETURN vorRLRef;
END SucheVorgaengerRL;


PROCEDURE DruckeRL(liste: RanglisteRef)=
(* Gibt die als Parameter gegebene Rangliste in absteigender Rangfolge aus *)

VAR ranglisteRef: RanglisteRef;
    rang        : CARDINAL;

BEGIN
  (* Gib Kopfkommentar der Rangliste aus *)
  SIO.PutLine(" *** Box-Rangliste *** ");
  SIO.Nl();

  ranglisteRef := liste;
  rang := 1;
  WHILE (ranglisteRef # NIL) DO

    (* Gib Rang und Daten des Boxers aus *)
    SIO.PutInt(rang); SIO.PutText(". ");
    SIO.PutText(ranglisteRef^.boxer.name.nachname & ", ");
    SIO.PutText(ranglisteRef^.boxer.name.vorname & "   ");
    SIO.PutText("Nation: ");
    SIO.PutText(ranglisteRef^.boxer.nation & "  ");
    SIO.PutText("Gewicht: ");
    SIO.PutInt(ranglisteRef^.boxer.gewicht);
    SIO.Nl();

    (* Inkrementiere Boxer und Rang *)
    ranglisteRef := ranglisteRef^.naechster;
    rang := rang + 1;

  END; (* WHILE *)
END DruckeRL;


PROCEDURE LoescheBoxerRL(VAR liste: RanglisteRef; boxer: Boxer)=
(* Loescht den gegebenen Boxer aus der Rangliste liste, falls vorhanden. *)

VAR ranglisteRef,
    vorgaengerRef: RanglisteRef;

BEGIN
  (* Suche den Boxer in der Liste *)
  ranglisteRef := SucheBoxerRL(liste, boxer);

  IF (ranglisteRef # NIL) THEN

    (* Vorhandenen Boxer aus der Rangliste loeschen *)
    vorgaengerRef := SucheVorgaengerRL(liste, boxer);
    IF vorgaengerRef = NIL THEN
      (* Erstes Element in Rangliste, daher kein Vorgaenger *)
      liste := ranglisteRef^.naechster;
    ELSE
      vorgaengerRef^.naechster := ranglisteRef^.naechster;
    END;
```

```
  ELSE
    SIO.PutLine("FEHLER: Boxer kommt nicht in der Liste vor!");
  END;
END LoescheBoxerRL;


PROCEDURE EinfuegenBoxerRL(VAR liste: RanglisteRef; boxer: Boxer)=
(* Fuegt den gegebenen Boxer am Ende der Rangliste liste ein *)

VAR ranglisteRef, aktRLRef: RanglisteRef;

BEGIN
  (* Pruefe, ob Boxer nicht schon in Rangliste enthalten *)
  ranglisteRef := SucheBoxerRL(liste, boxer);

  IF (ranglisteRef = NIL) THEN

    (* Erzeuge neues Ranglisten-Element fuer den Boxer *)
    ranglisteRef := NEW(RanglisteRef);
    ranglisteRef^.boxer := boxer;
    ranglisteRef^.naechster := NIL;

    (* Fuege Element an Ende der Liste an *)
    IF (liste = NIL) THEN
      (* Liste ist noch leer *)
      liste := ranglisteRef;
    ELSE
      (* Suche Ende der Liste *)
      aktRLRef := liste;
      WHILE (aktRLRef^.naechster # NIL) DO
        aktRLRef := aktRLRef^.naechster;
      END;
      (* Haenge Boxer ans Ende an *)
      aktRLRef^.naechster := ranglisteRef;
    END;
  ELSE
    SIO.PutLine("FEHLER: Boxer ist bereits in der Rangliste!");
  END;
END EinfuegenBoxerRL;


PROCEDURE AKommtHinterBInRL(aRef, bRef: RanglisteRef): BOOLEAN=
(* Prueft, ob das durch Zeiger aRef referenzierte Element hinter dem durch bRef
   referenzierten in der Rangliste vorkommt *)

VAR ranglisteRef: RanglisteRef;

BEGIN
  ranglisteRef := aRef;
  WHILE (ranglisteRef # NIL) AND (ranglisteRef # bRef) DO
    ranglisteRef := ranglisteRef^.naechster;
  END;

  RETURN NOT (ranglisteRef = bRef);
END AKommtHinterBInRL;


PROCEDURE AktualisiereRL(VAR liste: RanglisteRef; gewinner, verlierer: Boxer)=
(* Aendert die Rangfolge in der Rangliste entsprechend dem gegebenen
   Boxergebnis (Gewinner, Verlierer) *)

VAR gewinnerRef, verliererRef      : RanglisteRef;
    vorVerliererRef: RanglisteRef;
```

```
BEGIN
  (* Suche zunaechst Gewinner und Verlierer in der Rangliste *)
  gewinnerRef := SucheBoxerRL(liste, gewinner);
  verliererRef := SucheBoxerRL(liste, verlierer);

  IF (gewinnerRef = NIL) OR (verliererRef = NIL) THEN
    SIO.PutLine("FEHLER: Wenigstens einer der Boxer ist nicht in der
Rangliste!");
  ELSE

    (* Pruefe, ob nicht Gewinner ohnehin vor Verlierer in der Rangliste! *)
    IF NOT AKommtHinterBInRL(gewinnerRef, verliererRef) THEN
      SIO.PutLine("Keine Veraenderung in der Rangliste!");
    ELSE

      (* Fuege Verlierer hinter Gewinner in Rangliste ein *)

      (* Suche die jeweiligen Vorgaenger in der Rangliste *)
      vorVerliererRef := SucheVorgaengerRL(liste, verlierer);

      (* Entferne Verlierer aus ursprünglicher Position *)
      IF ( vorVerliererRef = NIL) THEN
        (* Kein Vorgaenger, da ganz am Anfang der Liste *)
        liste := verliererRef^.naechster;
      ELSE
        vorVerliererRef^.naechster := verliererRef^.naechster;
      END;

      (* Hänge Verlierer hinter Gewinner ein *)
      verliererRef^.naechster := gewinnerRef^.naechster;
      gewinnerRef^.naechster := verliererRef;
    END;
  END;
END AktualisiereRL;


PROCEDURE LeseBoxerEin(nachricht: TEXT): Boxer=
(* Fragt den Benutzer nach den Daten eines Boxers und gibt
   anschliessend einen entsprechenden Boxer zurueck *)

VAR resBoxer     : Boxer;
    dummy          : TEXT;

BEGIN
  SIO.PutLine(nachricht);
  SIO.PutText("Nachname   : ");
  resBoxer.name.nachname := SIO.GetLine();
  SIO.PutText("Vorname    : ");
  resBoxer.name.vorname := SIO.GetLine();
  SIO.PutText("Nation     : ");
  resBoxer.nation := SIO.GetLine();
  SIO.PutText("Gewicht    : ");
  resBoxer.gewicht := SIO.GetInt();

  (* Lies RETURN nach Gewicht aus der Eingabe! *)
  dummy := SIO.GetLine();

  RETURN resBoxer;
END LeseBoxerEin;

PROCEDURE DruckeMenue()=
(* Hilfsprozedur, die das Befehlsmenue ausgibt *)
BEGIN
  SIO.Nl();
```

```
  SIO.PutLine("Z : Zeige Rangliste an");
  SIO.PutLine("F : Fuege einen Boxer ein");
  SIO.PutLine("L : Loesche einen Boxer");
  SIO.PutLine("G : Kampf gemacht");
  SIO.PutLine("E : Exit");
  SIO.Nl();
END DruckeMenue;


BEGIN
  InitialisiereRL(rangliste);
  REPEAT
    DruckeMenue();

    SIO.PutText("Auswahl: ");
    auswahl := SIO.GetChar();
    dummy := SIO.GetLine(); (* Lies RETURN aus dem Eingabepuffer! *)
    SIO.Nl();

    CASE auswahl OF
    |  'Z', 'z' => DruckeRL(rangliste);
    |  'F', 'f' => boxer := LeseBoxerEin("Boxer Einfuegen"); SIO.Nl();
                   EinfuegenBoxerRL(rangliste, boxer);
    |  'L', 'l' => boxer := LeseBoxerEin("Boxer Loeschen"); SIO.Nl();
                   LoescheBoxerRL(rangliste, boxer);
    |  'G', 'g' => gewinner := LeseBoxerEin("Sieger des Spiels"); SIO.Nl();
                   verlierer := LeseBoxerEin("Verlierer des Spiels"); SIO.Nl();
                   AktualisiereRL(rangliste, gewinner, verlierer);
    |  'E', 'e' => (* Nichts machen, gleich ist ohnehin alles zu Ende! *)
    ELSE
      SIO.PutLine("Ungueltiger Befehl!");
    END (* CASE *)

  UNTIL (auswahl = 'e') OR (auswahl = 'E');
END Boxrangliste.
```