Aufgabe 1

Lösung: 12345

Aufgabe 2

Lösung:

```
TYPE SuperType = RECORD
                   SuperA: INTEGER;
                   CASE IsOne: BOOLEAN OF
                    | TRUE:
                      OneB  : CHAR;
                      OneC  : BOOLEAN;
                    | FALSE:
                      TwoA  : BITSET;
                      TwoC  : REAL;
                   END;
                 END;
```

Aufgabe 3

```
PROCEDURE SkipNoDigit(fp: FILE): BOOLEAN;
(* alle Zeichen, die keine Ziffern sind, ueberlesen -
   Position auf letztem Zeichen, das keine Ziffer ist *)

VAR
  ch: CHAR;
BEGIN
  WHILE Fgetc(ch, fp) & ~IsDigit(ch) DO
     ;
  END;
  IF ~Feof(fp) THEN
     IF ~Fungetc(ch, fp) THEN
       WriteString("Error: cannot unget char");
     END;
  END;
  RETURN Feof(fp);

END SkipNoDigit;
```

Aufgabe 4

Lösung: x = 7, y = 10

Aufgabe 5

Lösung:

a) 2 / 0 / 1 /

b) Fehler

c) 3 / 1 / 0 /

d) 0 / 0 / 0 /

e) 0 / 2 / 0 /

Aufgabe 9

Lösung:

a) x = 4, y = 0, z = 5, b = 2, w = 0.3

b) Fehler: Konstante b -> Var-Parameter

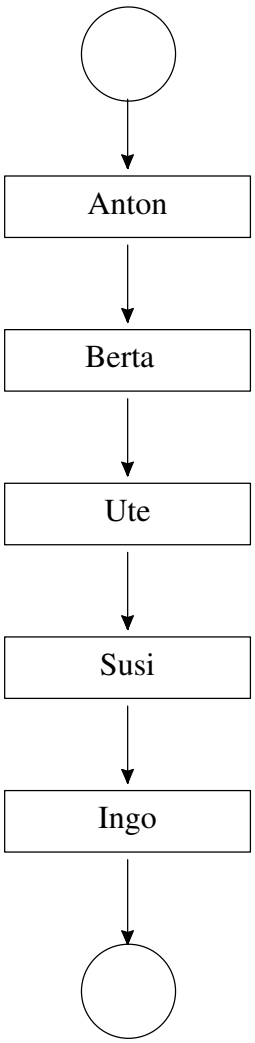
c) x = 4, y = 0, z = 5, b = 2, w = 0.3

d) x = 4, y = 0, z = 9, b = 2, w = 0.3

e) Fehler: Real -> Cardinal

Aufgabe 6

a)

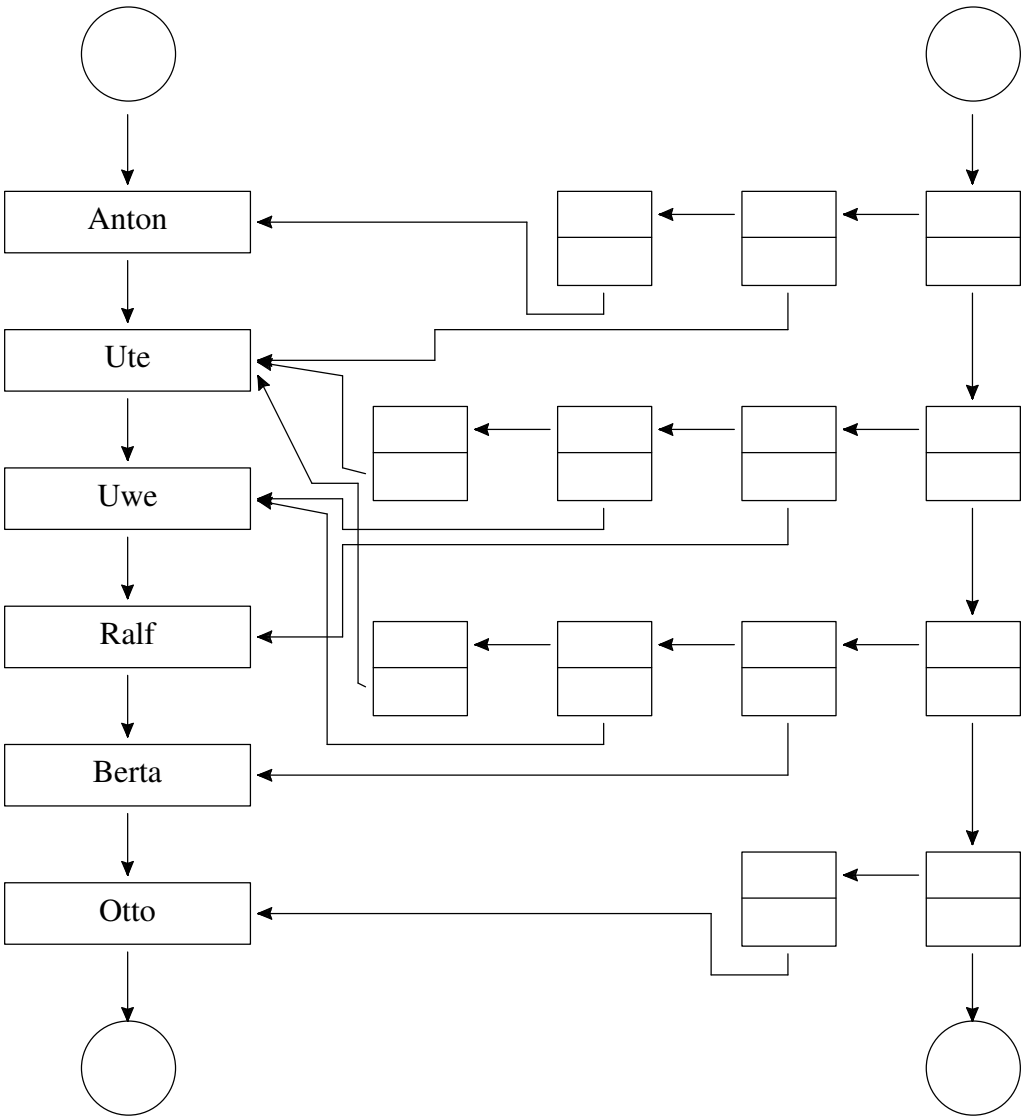


b)

```
TYPE
    MengePtr = POINTER TO Menge;
    Menge = RECORD
              wort: Wort;
              next: MengePtr;
            END;
```

Aufgabe 7

a)



b)

```
TYPE
   NamePtr = POINTER TO Name;  EineMePtr = POINTER TO EineMenge;
   Name = RECORD               EineMenge = RECORD
          wort: Wort;                 eineName: NamePtr;
          next: NamePtr;                nextName: EineMePtr;
         END;                        END;

   MengePtr = POINTER TO Mengen;
   Mengen = RECORD
             eineMenge: EineMePtr;
             nextMenge: MengePtr;
            END;
```

Aufgabe 8 MODULE sortBin; FROM InOut IMPORT Write, WriteLn, WriteString, WriteCard, ReadCard; FROM Storage IMPORT ALLOCATE;

```
CONST ANZ = 11;

TYPE    Baum =          POINTER TO Knoten;
      Knoten =        RECORD
                        inhalt: CARDINAL;
                        links, rechts: Baum;
                  END;

PROCEDURE baueBaum(a: ARRAY OF CARDINAL):Baum;
      PROCEDURE baue(uG, oG: INTEGER): Baum;
            VAR neuerKnoten: Baum;
            inh : CARDINAL;
            mitte: INTEGER;
      BEGIN
            IF uG > oG THEN RETURN NIL
            ELSE
                  mitte := uG + ((oG-uG+1) DIV 2);
                  inh := a[mitte];
                  NEW(neuerKnoten);
                  WITH neuerKnotenˆ DO
                        inhalt := inh;
                        links := baue(uG, mitte-1);
                        rechts := baue(mitte+1, oG)
                  END;
                  RETURN neuerKnoten;
            END
      END baue;
      VAR uG, oG: INTEGER; BEGIN
      uG := 0; oG := HIGH(a);
      RETURN baue(uG,oG) END baueBaum;
```