1. Aufgabe (5 Punkte)

Wie lautet die Ausgabe des folgenden Programms?

```
MODULE Example;

FROM InOut IMPORT WriteInt; FROM Storage IMPORT ALLOCATE, DEALLO-
CATE;

TYPE PointerToInt = POINTER TO INTEGER;

VAR
   IntPtr: PointerToInt;


 PROCEDURE Change(ip: PointerToInt);
 BEGIN
  ipˆ := 12345;
 END Change;


BEGIN
 NEW(IntPtr);
 IntPtrˆ := 1111;
 Change(IntPtr);
 WriteInt(IntPtrˆ, 1);
 DISPOSE(IntPtr); END Example.
```

2. Aufgabe (5 Punkte)

Fassen Sie die folgenden Typ - Vereinbarungen in einem "Super - Typ" (varianten Record) zusammen.

```
TYPE TypeOne = RECORD
          OneA: INTEGER;
          OneB: CHAR;
          OneC: BOOLEAN;
         END;


    TypeTwo = RECORD
          TwoA: BITSET;
          TwoB: INTEGER;
          TwoC: REAL;
         END;
```

3. Aufgabe (10 Punkte)

Schreiben Sie im nachfolgenden Teil die Funktion **SkipNoDigit**.

Hinweise:

- Verwenden Sie nur in der Vorgabe importierte Prozeduren oder die Funktion **IsDigit**.
- Verwenden Sie keine globalen Variablen.

```
DEFINITION MODULE Read;

FROM StdIO IMPORT FILE;

PROCEDURE ReadDigit(fp: FILE; VAR card: CARDINAL): BOOLEAN; (* aus Datei lesen - naechstes zu lesendes Zeichen ist Ziffer -
   lesen bis keine Ziffer mehr - Zahl konstruieren *)

PROCEDURE IsDigit(ch: CHAR): BOOLEAN;


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


END Read.

(* ------------------------------------------- *)

MODULE main;

FROM Read IMPORT SkipNoDigit, ReadDigit; FROM StdIO IMPORT Fopen, Feof, FILE, MODE, Fclose; FROM InOut IMPORT WriteString, WriteInt, WriteLn, Write; FROM SysExit IMPORT Exit;

VAR
  fp: FILE;
  digit: CARDINAL;
  result: BOOLEAN;

BEGIN
  IF ~Fopen(fp, 'test', read, TRUE) THEN        (* Datei 'test' *)
       WriteString("Error: cannot open file"); (* oeffnen *)
       Exit(1);
  END;
  WHILE ~Feof(fp) DO
    IF ~SkipNoDigit(fp) THEN
```

```
      result := ReadDigit(fp, digit);
      WriteInt(digit, 1); WriteLn;
   END;
END;
IF ~Fclose(fp) THEN ; END;
WriteLn; END main.
```

```
IMPLEMENTATION MODULE Read;

FROM StdIO IMPORT Fopen, Fdopen, Fread, Fwrite, Fseek, Ftell, Feof,
                  Ferror, Fgetc, Fputc, Fungetc, CloseAll, Fflush,
                  FILE, MODE, Fclose; FROM InOut IMPORT WriteString, WriteInt,
WriteLn, Write; FROM SysExit IMPORT Exit;

  (* Hier stehen die restlichen Prozeduren (ReadDigit, IsDigit),
     NICHT zu erstellen *)

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









  END SkipNoDigit;
```

END Read.

4. Aufgabe (5 Punkte)

Wie lautet die Ausgabe des folgenden Programms?

```
MODULE Obscure;

FROM InOut IMPORT WriteString, WriteInt, WriteLn; VAR
  x, y: CARDINAL;

  PROCEDURE Strange(VAR x: CARDINAL; y: CARDINAL): CARDINAL;
  BEGIN
    x := x + 1;
    RETURN (x + y);
  END Strange;

BEGIN
  x := 4;
  y := 2;
  x := Strange(x, y);
  y := Strange(y, x);
  WriteString("x = "); WriteInt(x, 1); WriteLn;
  WriteString("y = "); WriteInt(y, 1); WriteLn; END Obscure.
```

5. Aufgabe (10 Punkte)

Wie lautet die Ausgabe des folgenden Programmes (nach Abschluß der Einlesephase) bei Eingabe von:

a) 15 6 4 3 ?

b) -20 5 7 2 ?

c) 15 4 3 6 ?

d) c 5 3 2 ?

e) 10 14 5 7 ?

```
MODULE Zert;

FROM InOut IMPORT WriteLn, WriteString, WriteInt, ReadInt;

VAR lges, l1,l2, l3, z1,z2,z3: INTEGER;

PROCEDURE zerteilung(lges,l1,l2,l3: INTEGER): BOOLEAN; BEGIN
      IF lges = 0 THEN
            RETURN TRUE
      ELSIF lges < 0 THEN
            RETURN FALSE
      ELSE
            IF zerteilung(lges-l1,l1,l2,l3) THEN
                  INC(z1); RETURN TRUE;
            ELSIF zerteilung(lges-l2,l1,l2,l3) THEN
                  INC(z2); RETURN TRUE
            ELSIF zerteilung(lges-l3,l1,l2,l3) THEN
                  INC(z3); RETURN TRUE
            ELSE RETURN FALSE
            END
      END END zerteilung;

BEGIN
  z1 := 0; z2 := 0; z3 := 0;
  WriteString("lges = "); ReadInt(lges); WriteLn;
  WriteString("l1 = "); ReadInt(l1); WriteLn;
  WriteString("l2 = "); ReadInt(l2); WriteLn;
  WriteString("l3 = "); ReadInt(l3); WriteLn;   (* Ende Einlesen *)
  IF zerteilung(lges, l1,l2,l3) THEN
       WriteString("z1 / z2 / z3 = ");
       WriteInt(z1,4); WriteString(" / ");
       WriteInt(z2,4); WriteString(" / ");
       WriteInt(z3,4); WriteString(" / "); WriteLn;
  ELSE
        WriteString("Fehler"); WriteLn;
  END; END Zert.
```

6. Aufgabe (10 Punkte)

Gegeben sei folgende Typ-Vereinbarung:

```
TYPE Wort = ARRAY [0..MAX] OF CHAR
```

Geben Sie eine **dynamische** Datenstruktur an für die Repräsentation einer **Menge** von Elementen vom Typ *Wort*, und zwar

a) einmal in grafischer Darstellung für die konkrete Menge
M = {"anton", "berta", "ute", "susi", "ingo"}

b) und zum anderen als Typ-Definition(en) in

7. Aufgabe (10 Punkte)

Gegeben sei (wie in Aufgabe 6) folgende Typ-Vereinbarung:

```
TYPE Wort = ARRAY [0..MAX] OF CHAR
```

Geben Sie eine **dynamische** Datenstruktur für die Repräsentation einer **Menge von Mengen** von Elementen vom Typ *Wort*, so daß jedes elementare Element (vom Typ *Wort*) nur einmal gespeichert wird, und zwar

a) einmal in grafischer Darstellung fuer die konkrete Menge

```
K =   { {"anton", "ute"}, {"ute", "uwe", "ralf"},
        {"berta", "uwe", "ute"}, {"otto"}
      }
```

b) und zum anderen als Typ-Definition(en) in

8. Aufgabe (10 Punkte)

Schreiben Sie eine Prozedur **baueBaum**, die aus einem sortierten Feld einen nach Gewicht ausgeglichenen, inorder sortierten Baum aufbaut.

Hinweise:

Skript "Aufbau eines vollständig ausgeglichenen Binärbaumes" und binäres Sortieren.

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;
      Feld =          ARRAY[1..ANZ] OF CARDINAL;

PROCEDURE baueBaum(a: ARRAY OF CARDINAL):Baum; (* Voraussetzung: a
aufsteigend sortiert *)









END baueBaum;
```

9. Aufgabe (5 Punkte)

Gegeben sei folgendes Programm:

```
MODULE Try;

CONST
  b = 2;

VAR
  x, y, z : CARDINAL;
  w : REAL;

  PROCEDURE Ex(i, y: CARDINAL; VAR j: CARDINAL);
  BEGIN
      i := 5;
      j := i + y;
      y := y + 1;
  END Ex;

BEGIN
  w := 0.3;
  x := 4;
  y := 0;
  z := 3;

  (* ********** *)

END Try.
```

An die gekennzeichnete Stelle im Hauptprogramm soll jeweils einer der folgenden Aufrufe von **Ex** gesetzt werden. Geben Sie an, welche Werte *x, y, z, b, w* nach dem Aufruf von *Ex* jeweils haben. Sollten einige der Aufrufe nicht zulässig sein, geben Sie an weshalb nicht.

a) Ex(b, y, z);

b) Ex(x, y, b);

c) Ex(z, y, z);

d) Ex(x, y+x, z);

e) Ex(x,w,y);