

Name:				
Vorname:				
Matrikelnummer:				
Punkte:	/24 =	%	Note:	

Klausur im SS 95

zur

Informatik II

am 20. Juni 1995

Hinweise:

- Bitte jedes Blatt einzeln mit Ihrem Namen, Ihrer Matrikelnummer und der Nummer der bearbeiteten Aufgabe bezeichnen.
- Als Hilfsmittel sind **beliebige schriftliche Unterlagen** zugelassen, jedoch **keine Rechner**. Klausurabgabetermin ist 15³⁰ Uhr.
- Sie können farbige Stifte verwenden, jedoch **kein Rot**.
- Alle Fragen beziehen sich auf die der Vorlesung und dem Labor zugrundegelegte Sprache Oberon-2.
- Die Klausur besteht aus insgesamt 5 Aufgaben.

Aufgabe 1 (auf gesondertem Blatt)

(Punktzahl: 4/24)

Ein Suchrätsel: Die folgenden zwei Module enthalten eine Reihe von Fehlern, die Sie korrigieren sollen. Änderungen, die nicht nur der Fehlerkorrektur dienen, dürfen nicht vorgenommen werden. Schreiben Sie beide Module vollständig neu auf; fügen Sie keine Änderungen in den vorgedruckten Text ein.

```

MODULE KlausurAufgabe;
  IMPORT In, Out;
  CONST f = 12, N = 4;
  TYPE Vector* = POINTER TO VectorDesc;
  VectorDesc = ARRAY
    N OF REAL
  END;

  PROCEDURE CreateVector*(v: Vector);
  BEGIN NEW(v) END CreateVector;

  PROCEDURE SetVals*(v);
    VAR i: INTEGER;
  BEGIN FOR i := 0 TO N-1 DO In.Real(v[i]) END
  END SetVals;

  PROCEDURE ShowVals(v: Vector);
    VAR i: INTEGER;
  BEGIN FOR i := 0 TO N-1 DO Out.Real(v[i], f) END
  END ShowVals;

  PROCEDURE Add*(u, v: Vector): Vector;
    VAR w: Vector;
  BEGIN CreateVector(w); w^ := u^ + v^; RETURN w
  END Add;

END KlausurAufgabe.

```

```

MODULE KlausurAufgabeTest;
  IMPORT In, Out, K := KlausurAufgabe;

  PROCEDURE Start*;
    VAR u, v, w: Vector;
  BEGIN In.Open;
    K.CreateVector(u); K.CreateVector(v);
    K.SetVals(u); K.SetVals(v);
    K.ShowVals(Add(u, v)); Out.Ln
  END Start;

END KlausurAufgabeTest.

KlausurAufgabeTest.Start
1.0 2.0 3.0 4.0
2.0 3.0 4.0 5.0 ~

```

Aufgabe 2

(Punktzahl: 4/24)

Das Spiel *Türme von Hanoi* gilt als Standardbeispiel für ein durch Rekursion besonders elegant lösbares Problem. Das folgende Modul gibt die Struktur wieder, ohne daß die Prozedur **ZieheScheibe** im Einzelnen ausgeführt wäre. Statt dessen ist eine globale Variable **z** eingeführt, deren Wertänderungen einen Einblick in die aufeinanderfolgenden rekursiven Prozeduraufrufe gestatten.

Welche Werte von **z** werden nacheinander ausgegeben, wenn die Prozedur **Start** mit dem angegebenen Wert **N = 3** gestartet wird?

```
MODULE KlausurAufgabe;
IMPORT In, Out;
VAR z, N: INTEGER;

PROCEDURE ZieheScheibe(Ausgang, Ziel: INTEGER);
BEGIN (* ... Scheibe ziehen ... *)
END ZieheScheibe;

PROCEDURE Hanoi(N, A, B, C: INTEGER);
BEGIN INC(z); Out.Int(z, 5);
  IF N = 1 THEN ZieheScheibe(A, C)
  ELSE
    Hanoi(N-1, A, C, B);
    ZieheScheibe(A, C);
    Hanoi(N-1, B, A, C)
  END;
  DEC(z)
END Hanoi;

PROCEDURE Start*;
BEGIN In.Open; In.Int(N); Hanoi(N, 1, 2, 3); Out.Ln
END Start;

BEGIN z := 0 END KlausurAufgabe.

KlausurAufgabe.Start 3 ~
```

ausgegebene Werte:									
--------------------	--	--	--	--	--	--	--	--	--

(Die Tabelle gibt nicht die Anzahl der ausgegebenen Werte richtig wieder; diese zu finden bleibt Ihnen überlassen.)

Aufgabe 3 (auf gesondertem Blatt)

(Punktzahl: 5/24)

In dem folgenden Modul wird eine Liste von Studierenden bearbeitet. Die Prozedur **Init** soll hier keine weitere Rolle spielen. Es wird nur unterstellt, daß die Tabelle **S** durch **Init** vollständig ausgefüllt wurde. Leere Einträge kommen also nicht vor. Schreiben Sie unter Verwendung einer doppelten **FOR**-Schleife eine Prozedur **PrintAll**, die die gesamte Tabelle ausdrückt. Jeder Eintrag ist in einer Zeile in der Reihenfolge

SemNr LfdNr Nachname Vorname Alter auszugeben. Abgesehen von dieser Festlegung sollen die genauen Ausgabeformate keine Rolle spielen.

Die Auflistung soll so erfolgen, daß für ein gegebenes Semester (**SemNr**) stets alle laufenden Nummern (**LfdNr**) ausgegeben werden bevor die nächste **SemNr** folgt.

Beachten Sie die Zählweise: das 1., 2., ... Semester wird als Nummer 1, 2, ... angesprochen.

```
MODULE KlausurAufgabe;
IMPORT Out;
CONST MaxSemNr = 15; MaxLfdNr = 100;
TYPE
  SemesterNr = INTEGER; LaufendeNr = INTEGER;
  Name = ARRAY 20 OF CHAR;
  Student = POINTER TO StudentDesc;
  StudentDesc = RECORD
    Vorname, Nachname: Name;
    Alter: INTEGER
  END;
  Array = ARRAY MaxSemNr + 1, MaxLfdNr + 1 OF Student;
  (* Numerierung von 1 ... MaxSemNr bzw. 1 ... MaxLfdNr *)

PROCEDURE Init(VAR S: Array);
(* ... Init belegt das gesamte S mit Werten ...
   Einzelheiten sollen hier keine Rolle spielen ... *)
END Init;

PROCEDURE PrintAll(S: Array);
  VAR SemNr: SemesterNr; LfdNr: LaufendeNr;
BEGIN
  (* ... diese Prozedur soll ausgearbeitet werden ... *)
END PrintAll;

PROCEDURE Start*;
  VAR S: Array;
BEGIN Init(S); PrintAll(S) END Start;

END KlausurAufgabe.
```

Aufgabe 4

(Punktzahl: 6/24)

Welche Werte werden von dem nebenstehend abgedruckten Modul beim Start mit Neuladen und bei einem unmittelbar nachfolgenden Start ohne Neuladen ausgegeben?

```
MODULE KlausurAufgabe;  
  IMPORT Out;  
  VAR z: INTEGER;  
  
  PROCEDURE P(VAR y, x: INTEGER);  
  BEGIN y := z; x := y + z END P;  
  
  PROCEDURE Q(VAR x: INTEGER; y: INTEGER);  
  BEGIN INC(z); P(z, x); z := y - z END Q;  
  
  PROCEDURE ShowVals(x, y, z: INTEGER);  
  BEGIN Out.Int(x, 5); Out.Int(y, 5); Out.Int(z, 5); Out.Ln  
  END ShowVals;  
  
  PROCEDURE Start*;  
  VAR x, y: INTEGER;  
  BEGIN  
    x := 1; y := 1; P(x, y); Q(x, y); ShowVals(x, y, z);  
    x := 1; y := 1; P(x, y); Q(x, y); ShowVals(x, y, z);  
    x := 1; y := 1; P(x, y); Q(x, y); ShowVals(x, y, z)  
  END Start;  
  
BEGIN z := 2 END KlausurAufgabe.
```

nach Aufruf von KlausurAufgabe.Start:	x	y	z
Werte mit Neuladen des Moduls			
Werte nach erneutem Aufruf ohne Neuladen des Moduls			

Aufgabe 5

(Punktzahl: 5/24)

Kreuzen Sie an(×), ob die folgenden Oberon-2-Konstrukte jeweils syntaktisch richtig oder falsch sind:

	richtig	falsch
VAR IN: INTEGER;		
VAR LI: LONGINT;		
VAR p: POINTER TO REAL;		
VAR q: POINTER TO RECORD END;		
VAR A: ARRAY 1,2,3,4 OF INTEGER;		
VAR B: ARRAY 1.0, 5.0 OF REAL;		
VAR r: POINTER TO POINTER TO RECORD R: REAL END;		
VAR R: RECORD x: REAL; i: INTEGER END;		
TYPE Array = ARRAY 1, 2 OF REAL; VAR x,y,z: Array;		
TYPE Record = POINTER TO RECORD END; VAR P: POINTER TO Record;		