

Hinweise:

- Es sind beliebige schriftliche Unterlagen als Hilfsmittel zugelassen, jedoch **keine Rechner**, auch **keine Taschenrechner**.
- Die Klausur besteht aus 4 Aufgaben. Klausurabgabetermin ist 16³⁰ Uhr.
- Analysieren Sie die abgedruckten Programme und beantworten Sie dann die gestellten Fragen; füllen Sie die Kästchen auf dieser Seite aus, jeweils mit einer Zahl pro Kästchen – gegebenenfalls mit Vorzeichen – oder – im Falle von Buchstaben – einem Zeichen pro Kästchen ohne Rücksicht auf eventuell auftretende Zwischenräume. Die Anzahl der vorgedruckten Kästchen muß nicht gleich der Zahl der einzutragenden Ergebnisse sein.
- **WICHTIG:** Bewertet werden nur die Ergebnisse, die Sie auf diesem ersten Blatt eingetragen haben; fehlende Eintragungen an dieser Stelle werden stets so gewertet wie fehlerhafte Eintragungen!

Name:
Vorname:
Mat.-Nr.:
erreichte Punktzahl/Note:

Aufgabe 1 (Punktzahl: 7/24)

a)

--	--	--	--	--	--	--	--	--

b)

--	--	--	--	--	--	--	--	--

c)

--

d)

--	--	--	--	--	--	--	--	--

e)

Aufgabe 2 (Punktzahl: 6/24)

a)

--	--	--	--	--	--	--	--	--

b)

--	--	--	--	--	--	--	--	--

c)

--	--	--	--	--	--	--	--	--

d)

--	--	--	--	--	--	--	--	--

Aufgabe 3 (Punktzahl: 6/24)

a)

--	--	--	--	--	--	--	--	--

b)

--	--	--	--	--	--	--	--	--

c)

--	--	--	--	--	--	--	--	--

d)

--	--	--	--	--	--	--	--	--

Aufgabe 4 (Punktzahl: 5/24)

[illegible]

Das folgende Programm ist eine abstrahierte und etwas modifizierte Fassung des Beispiels *Grafik-Editor* aus der Vorlesung. Machen Sie sich die Struktur des Programms klar, und beantworten Sie dann die unten gestellten Fragen zu einigen Details des Programms. Beachten Sie, dass das Modul *IntItems* noch einige Fehler enthält, auf die sich die erste der Fragen bezieht.

```

MODULE BasicStructure;
IMPORT Out;
CONST fmt = 15; N = 100;
TYPE
  Message* = RECORD END;
  PrintMsg* = RECORD (Message) END;
  CalculateMsg* = RECORD (Message)
    value*: INTEGER
  END;

  Item* = POINTER TO ItemDesc;
  Array = ARRAY OF Item;

  Handler* = PROCEDURE (q: Item;
                        VAR msg: Message);

  ItemDesc* = RECORD
    handle*: Handler
  END;

  Struct* = POINTER TO StructDesc;
  StructDesc = RECORD
    list: ARRAY N OF Item;
    p: INTEGER
  END;

VAR
  newItem*: PROCEDURE (): Item;
  Z: INTEGER; (* Zähler für Aufrufe von Swap *)

PROCEDURE Insert(S: Struct; q: Item);
BEGIN
  IF S.p < N THEN S.list[S.p] := q; INC(S.p) END
END Insert;

PROCEDURE CreateItem*(S: Struct);
  VAR q: Item; print: PrintMsg;
BEGIN
  q := newItem(); Insert(S, q)
END CreateItem;

PROCEDURE ShowObjects*(S: Struct);
  VAR q: Item; print: PrintMsg; i: INTEGER;
BEGIN
  FOR i := 0 TO S.p - 1 DO
    q := S.list[i];
    q.handle(q, print)
  END
END ShowObjects;

PROCEDURE PrintValues*(S: Struct);
  VAR q: Item; i: INTEGER; calc: CalculateMsg;
BEGIN
  FOR i := 0 TO S.p - 1 DO
    q := S.list[i];
    q.handle(q, calc);

```

```

    Out.Int(calc.value, fmt)
  END;
  Out.Ln
END PrintValues;

PROCEDURE getval(VAR a: Array; i: INTEGER);
  INTEGER;
  VAR calc: CalculateMsg;
BEGIN
  a[i].handle(a[i], calc);
  RETURN calc.value
END getval;

PROCEDURE Swap(VAR a: Array; i: INTEGER);
  VAR x: Item;
BEGIN
  INC(Z);
  x := a[i]; a[i] := a[i-1]; a[i-1] := x
END Swap;

PROCEDURE Proc*(S: Struct);
  PROCEDURE proc(VAR a: Array);
    VAR i, j, lo, up: INTEGER;
  BEGIN
    Z := 0;
    lo := 0; up := S.p - 1;
    WHILE up > lo DO
      j := lo;
      FOR i := lo TO up-1 DO
        IF getval(a, i) > getval(a, i+1) THEN
          Swap(a, i+1); j := i
        END
      END; (* of FOR *)
      up := j;
      ShowObjects(S); Out.Ln; (* ← nur zu Frage e *)
      FOR i := up TO lo+1 BY -1 DO
        IF getval(a, i) < getval(a, i-1) THEN
          Swap(a, i); j := i
        END
      END; (* of FOR *)
      lo := j
    END (* of WHILE *)
  END proc;
BEGIN
  proc(S.list); Out.Int(Z, fmt); Out.Ln
END Proc;

PROCEDURE Open*(VAR S: Struct);
  BEGIN NEW(S);
    S.p := 0;
  END Open;

END BasicStructure.

```

```

MODULE IntItems;
IMPORT BS := BasicStructure, In, Out;
CONST fmt = 8; a = 2; b = 1;
TYPE
1   IntItem = POINTER TO IntItemDesc;
2   IntItemDesc = RECORD (BS.ItemDesc)
3       x: INTEGER
4   END;

5   PROCEDURE HandleIntItem(q: BS.Item;
    (* ... noch Zeile 5 *) VAR msg: BS.Message);
6   BEGIN
7       WITH q: IntItem DO
8           IF msg IS BS.PrintMsg THEN
9               Out.Int(q.x, fmt)
10          ELIF msg IS BS.CalculateMsg THEN
11              msg.value := a*q.x + b
12          ELSE
13              Out.String("unknown message: no action ... ");
14              Out.Ln
15          END
16      END
17  END HandleIntItem;

18  PROCEDURE New(): Item;
19      VAR int: IntItem;
20      BEGIN NEW(int);
21      int.handle := HandleIntItem;
22      In.Int(int);
23      RETURN int
24  END New;

25  PROCEDURE Set*;
26  BEGIN BS.newItem := New()
    END Set;

END IntItems.

```

```

MODULE Application;
IMPORT BS := BasicStructure, In, Out;

VAR S: BS.Struct;

PROCEDURE CreateItem*;
BEGIN
    In.Open;
    BS.CreateItem(S);
END CreateItem;

PROCEDURE ShowObjects*;
BEGIN
    BS.ShowObjects(S); Out.Ln
END ShowObjects;

PROCEDURE PrintValues*;
BEGIN
    BS.PrintValues(S); Out.Ln
END PrintValues;

PROCEDURE Proc*;
BEGIN
    BS.Proc(S)
END Proc;

PROCEDURE Open*;
BEGIN
    BS.Open(S)
END Open;

BEGIN BS.Open(S)
END Application.

```

- a) Im Modul **IntItems** sind einige Syntaxfehler, die der Compiler festgestellt hat, noch nicht korrigiert. Geben Sie die Zeilennummern an, in denen sich diese Fehler befinden. Gefragt wird also ausdrücklich nur nach den Zeilennummern, nicht nach einer expliziten Korrektur!
- Im folgenden werde unterstellt, dass diese Fehler beseitigt sind.
- b) Es mögen Objekte vom Typ **IntItem** erzeugt und dabei – in dieser Reihenfolge – mit den Werten 2, -4, -1, 5, -7, 3 belegt werden. Wie lautet dann die Ausgabe nach einer Aktivierung von **Application.PrintValues**?
- c) Wie häufig wird mit diesen Eingabewerten durch eine Aktivierung von **Application.Proc** die Prozedur **Swap** aufgerufen (Wert des Zählers **Z** am Ende)?
- d) Wie lautet die Ausgabe, wenn nach Aktivierung von **Application.Proc** erneut **Application.PrintValues** aufgerufen wird?
- e) Das Programm werde erneut initialisiert mit den gleichen Eingabewerten wie oben. Welches sind dann die Arraybelegungen (so wie angezeigt durch **ShowObjects**) beim Durchlaufen der durch „ $\Leftarrow$ “ markierten Stelle, wenn **Application.Proc** aktiviert wird?

```

MODULE KlausurAufgabe2;
IMPORT In, Out;
TYPE
  TypeKey = INTEGER;
  List = POINTER TO ListDesc;
  ListDesc = RECORD
    k: TypeKey;
    next: List;
  END;

PROCEDURE search(key: TypeKey; VAR head: List):
  List;
  VAR p, q: List;
BEGIN
  IF head = NIL THEN RETURN NIL
  ELSIF key = head.k THEN RETURN head
  ELSE
    p := head;
    WHILE p.next # NIL DO
      IF p.next.k = key THEN
        q := head;
        head := p.next;
        p.next := p.next.next;
        head.next := q;
        RETURN head
      ELSE
        p := p.next
      END
    END
  END;
  RETURN NIL
END search;

PROCEDURE Insert(VAR head: List; key: TypeKey);
  VAR p: List;
BEGIN
  NEW(p); p.k := key; p.next := head;
  head := p
END Insert;

```

```

PROCEDURE Show(head: List);
BEGIN
  IF head = NIL THEN
    Out.String("leer")
  ELSE
    WHILE head # NIL DO
      Out.Int(head.k, 5);
      head := head.next
    END
  END;
  Out.Ln
END Show;

PROCEDURE Start*;
  VAR x: TypeKey; p, q: List;
BEGIN
  p := NIL; q := NIL;
  Show(p); (* <== ?? Frage a) *)

  In.Open; In.Int(x);
  WHILE In.Done DO Insert(p, x); In.Int(x) END;
  Show(p); (* <== ?? Frage b) *)

  q := search(7, p); q := search(3, p);
  Show(p); (* <== ?? Frage c) *)

  q := search(2, p);
  Show(p); (* <== ?? Frage d) *)

END Start;
END KlausurAufgabe2.

```

KlausurAufgabe2.Start 5 2 7 4 9 10 ~

Was gibt das abgedruckte Programm jeweils aus an den mit a), b), c) und d) markierten Stellen, wenn es mit den am Ende des Programms angegebenen Werten gestartet wird?

```

MODULE KlausurAufgabe3a;
IMPORT In, Out, Stack;
CONST SP = 20X;

PROCEDURE ReadChar(VAR c: CHAR);
(* nächstes Zeichen > SP bereitstellen *)
BEGIN In.Char(c);
  WHILE In.Done & (c <= SP) DO In.Char(c) END
END ReadChar;

PROCEDURE Do*;
VAR c: CHAR; x: INTEGER;
BEGIN
  Stack.Init;
  In.Open; ReadChar(c);
  WHILE In.Done & (c # '~') DO
    CASE c OF
      '*': x := Stack.Pop() * Stack.Pop();
      '+': x := Stack.Pop() + Stack.Pop();
      '0'..'9':
        x := 0;
        REPEAT
          x := 10*x + (ORD(c) - ORD('0')); In.Char(c)
        UNTIL (c < '0') OR ('9' < c)
      ELSE Out.String("Fehler"); Out.Ln; HALT(31)
    END;
    Stack.Push(x);
    ReadChar(c)
  END;

  Out.Int(Stack.Pop(), 8); Out.Ln
END Do;

END KlausurAufgabe3a.

```

```

MODULE KlausurAufgabe3b;
IMPORT In, Out, Stack;
CONST SP = 20X;

PROCEDURE ReadChar(VAR c: CHAR);
... wie in Modul KlausurAufgabe3a
END ReadChar;

PROCEDURE Do*;
VAR c: CHAR;
BEGIN
  Stack.Init;
  In.Open; ReadChar(c);
  WHILE In.Done & (c # '~') DO
    CASE c OF
      ')': Out.Char(CHR(Stack.Pop())); Out.Char(SP)
      '+': Stack.Push(ORD(c))
      '*': Stack.Push(ORD(c))
      '0'..'9':
        REPEAT
          Out.Char(c); In.Char(c)
        UNTIL (c < '0') OR ('9' < c);
        Out.Char(SP)
      ELSE
        IF c # '(' THEN Out.Char(SP) END
      END;
      ReadChar(c)
    END;
    Out.Ln
  END Do;

END KlausurAufgabe3b.

```

Was ist die Ausgabe der abgedruckten Programme jeweils nach Aktivierung von

- KlausurAufgabe3a.Do 1 3 + 5 6 + * 2 * 3 + ~
- KlausurAufgabe3a.Do 1 13 * 13 - ~
- KlausurAufgabe3b.Do (((13 + 5) * 7) + 2) ~
- KlausurAufgabe3b.Do (((3 + 5) * (2 + 2)) * 7) ~

Hinweise:

- Die Prozeduren **Push** und **Pop** des importierten Moduls **Stack** haben ihre normale Bedeutung; **Init** ist erforderlich für die Lauffähigkeit des Programms, spielt für die Beantwortung der Fragen aber keine Rolle.
- Sofern eines der Programme in einen Trap läuft, so geben Sie *nur* den Text an, der *unmittelbar vorher* noch ausgegeben wird, sonst nichts weiter.
- Für die hier gestellten Fragen spielt es keine Rolle, dass die Programme keine Syntaxprüfung der Eingabestrings vornehmen.

```

MODULE KlausurAufgabe4;
  CONST N = 4;

  TYPE
    OArray = ARRAY OF INTEGER;
    SArray = ARRAY N OF INTEGER;

    Item = POINTER TO ItemDesc;
    ItemDesc = RECORD
      END;

    IntItem = POINTER TO IntItemDesc;
    IntItemDesc = RECORD(ItemDesc)
      X: INTEGER
    END;

    RealFunction = PROCEDURE (x: REAL): REAL;
    StringProc = PROCEDURE(s: ARRAY N OF CHAR);

  VAR
    p: Item;
    q: IntItem;
    f: RealFunction;
    g: StringProc;
    a: SArray;
    b: ARRAY N OF INTEGER;

    (* ..... *)
    PROCEDURE P(A, B: OArray);          (* a *)
    BEGIN
      A := B
    END P;
    (* ..... *)

    (* ..... *)
    PROCEDURE (VAR p: IntItem) Q;      (* b *)
    BEGIN
      p.X := 123
    END Q;
    (* ..... *)

    (* ..... *)
    PROCEDURE R(VAR y: ItemDesc);      (* c *)
    VAR x: IntItemDesc;
    BEGIN
      y := x
    END R;
    (* ..... *)

```

```

PROCEDURE S(A: SArray);
BEGIN
END S;

PROCEDURE F(x: REAL): REAL;
BEGIN
  RETURN x*x
END F;

PROCEDURE Test*;
  VAR x: INTEGER; y: LONGREAL;

  (* ..... *)
  PROCEDURE (VAR y: IntItemDesc) T;   (* d *)
  BEGIN
    y.X := 123
  END T;
  (* ..... *)

  BEGIN
    (* ..... *)
    y := x;                            (* e *)

    (* ..... *)
    f := F;                            (* f *)

    (* ..... *)
    S(a);                              (* g *)

    (* ..... *)
    S(b);                              (* h *)

    (* ..... *)
    p := q;                            (* i *)

    (* ..... *)
    g("abcd")                         (* j *)

    (* ..... *)

  END Test;
END KlausurAufgabe4.

```

In dem abgedruckten Programm sind einige Abschnitte markiert als a, b, ..., j. In welchen dieser Abschnitte *gibt es einen Fehler* zur Compilezeit (Antwort: **0**), in welchen der Abschnitte *gibt es keinen solchen Fehler* (Antwort: **1**)? Beachten Sie: gefragt ist ausdrücklich nach *Compilezeitfehlern*, die Lauffähigkeit des Programms steht nicht zur Diskussion.

Hinweise zur Bewertung:

- Fehlenden Antworten werden als Fehler gewertet.
- *Ein* Fehler wird nicht negativ gewertet; für jeden weiteren Fehler wird ein Punkt abgezogen, d.h., insgesamt 6 oder mehr Fehler ergeben 0 Punkte.