

[illegible]

In der Prozedur **Algorithm** des folgenden Programmszugs ist ein bekannter Sortieralgorithmus implementiert. Sie dient als Basis für diese Aufgabe.

<pre> MODULE KlausurAufgabe2; ... VAR a: ARRAY N OF INTEGER; PROCEDURE Algorithm(L, R: INTEGER); VAR i, j: INTEGER; w, x: INTEGER; BEGIN i := L; j := R; x := a[(L+R) DIV 2]; REPEAT WHILE a[i] < x DO INC(i) END; WHILE x < a[j] DO DEC(j) END; </pre>	<pre> IF i <= j THEN w := a[i]; a[i] := a[j]; a[j] := w; INC(i); DEC(j) END UNTIL i > j; IF L < j THEN Algorithm(L, j) END; IF i < R THEN Algorithm(i, R) END; END Algorithm; ... END KlausurAufgabe2. </pre>
--	---

Vervollständigen Sie in dem folgenden Text die markierten Stellen, so daß das Programm durch das Kommando **KlausurAufgabe2b.Start** gestartet werden und ohne Laufzeitfehler ablaufen kann. Die Prozedur **Algorithm** im Modul **KlausurAufgabe2b** soll – bezogen auf **Item** – Entsprechendes leisten wie die gleichnamige Prozedur in dem oben abgedruckten Modul – bezogen auf **INTEGER**.

- Schreiben Sie in dieser Teilaufgabe den mit (a) markierten fehlenden Programmtext auf.
- Schreiben Sie in dieser Teilaufgabe die Prozedur **Algorithm** (Markierung (b)) vollständig auf, also mit Prozedurkopf und dem abschließend **END Algorithm**.

<pre> MODULE KlausurAufgabe2a; IMPORT K := KlausurAufgabe2b, In, Out; TYPE Item = POINTER TO ItemDesc; ItemDesc = RECORD (K.ItemDesc) key: ARRAY 64 OF CHAR END; ... bitte vervollständigen ... (a) ⇐ END KlausurAufgabe2a. MODULE KlausurAufgabe2b; IMPORT In, Out; CONST N = 10; TYPE Item* = POINTER TO ItemDesc; ItemDesc* = RECORD END; VAR a: ARRAY N OF Item; newItem*: PROCEDURE(): Item; PROCEDURE (x: Item) LessThan*(y: Item); BOOLEAN; BEGIN HALT(21) END LessThan; PROCEDURE (x: Item) Read*; BEGIN HALT(22) END Read; PROCEDURE (x: Item) Print*; BEGIN HALT(23) END Print; </pre>	<pre> PROCEDURE Algorithm(L, R: INTEGER); ... bitte vervollständigen ... (b) ⇐ END Algorithm; PROCEDURE Show*; VAR i: INTEGER; BEGIN i := 0; WHILE (i <= N-1) & (a[i] # NIL) DO a[i].Print; INC(i) END; Out.Ln END Show; PROCEDURE Start*; VAR n: INTEGER; x: Item; BEGIN In.Open; n := 0; x := newItem(); WHILE (n <= N-1) & In.Done DO a[n] := x; INC(n); x := newItem() END; Show; IF n > 1 THEN Algorithm(0, n-1) ELSE Out.String("error"); Out.Ln END; Show END Start; END KlausurAufgabe2b. </pre>
---	--

- a) Tragen Sie in die unten gedruckte Tabelle ein, welche Werte durch das folgende Programm ausgegeben werden (jeweils ein Zahlenwert pro Feld). Zeilenvorschübe sind zu berücksichtigen, Formate innerhalb eines Feldes (Konstante `fmt`) jedoch nicht. Zur Vereinfachung sind die Stellen markiert, an denen eine Ausgabe erfolgt.

Die vorgedruckte Tabelle ist nach unten und nach rechts offen. Es ist Teil der Aufgabe herauszufinden, wieviele Felder in den Zeilen und Spalten wirklich auszufüllen sind.

```

MODULE KlausurAufgabe3;
IMPORT In, Out;
CONST N = 100; fmt = 5;
TYPE Item = INTEGER;
VAR a: ARRAY N OF Item;

PROCEDURE Min(a, b: Item): Item;
BEGIN
  IF a <= b THEN RETURN a
  ELSE RETURN b
  END
END Min;

PROCEDURE Max(a, b: Item): Item;
BEGIN
  IF a >= b THEN RETURN a
  ELSE RETURN b
  END
END Max;

PROCEDURE Write(k: Item);
BEGIN Out.Int(k, fmt)
END Write;

PROCEDURE Read(VAR k: Item);
BEGIN In.Int(k)
END Read;

PROCEDURE ShowArray(i, j: INTEGER);
BEGIN
  WHILE i <= j DO Write(a[i]); INC(i) END;
  Out.Ln
END ShowArray;

```

```

PROCEDURE Q(i, k: INTEGER; VAR X, Y: Item);
  VAR m1, m2, m3, m4: Item;
BEGIN
  ShowArray(i, k);
  IF i = k THEN
    X := a[i]; Y := a[i]
  ELSIF k - i = 1 THEN
    X := Min(a[i], a[k]);
    Y := Max(a[i], a[k])
  ELSE
    Q(i, (i+k) DIV 2, m1, m2);
    Q(1 + (i+k) DIV 2, k, m3, m4);
    X := Min(m1, m3);
    Y := Max(m2, m4)
  END
END Q;

PROCEDURE Start*;
  VAR n: INTEGER; x, X, Y: Item;
BEGIN In.Open;
  n := 0; Read(x);
  WHILE (n <= N-1) & In.Done DO
    a[n] := x; INC(n); Read(x);
  END;
  IF n >= 1 THEN
    Q(0, n-1, X, Y);
    Write(X); Write(Y)
  END
END Start;

END KlausurAufgabe3.

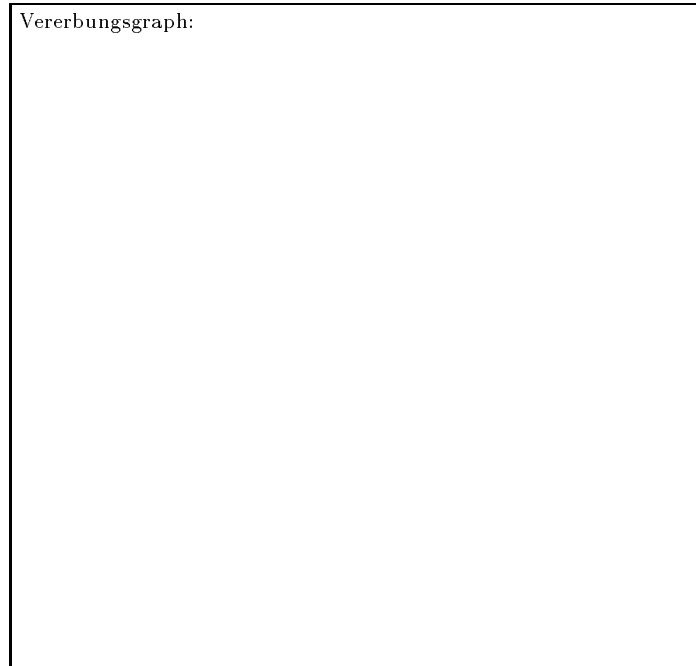
```

KlausurAufgabe3.Start 1 9 3 5 2 ~

- b) Beschreiben Sie in einem Satz – ohne Begründung oder weitere Erläuterungen –, welches die allgemeine Wirkung der Prozedur `Q` ist, wobei der Aufruf von `ShowArray` innerhalb `Q` nicht zu berücksichtigen ist:

Ein Prozeduraufruf `Q(i, k, X, Y) ...`

- a) Zeichnen Sie den Vererbungsgraphen der Datenstruktur in dem nachfolgend abgedruckten Modul mit den üblichen Konventionen: speziellere Klasse nach unten, gerichtete Kante von der spezielleren zur allgemeineren Klasse. Verwenden Sie zur Bezeichnung der Klassen in dem Graphen die Namen der Zeigertypen, also A, B, ...



- b) Welches ist die Ausgabe nach einer Kommandoaktivierung `KlausurAufgabe4.Start`?

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

```

MODULE KlausurAufgabe4;
  IMPORT Out;
  TYPE
    A = POINTER TO ADesc;
    ADesc = RECORD END;

    B = POINTER TO BDesc;
    BDesc = RECORD (ADesc) END;

    C = POINTER TO CDesc;
    CDesc = RECORD (BDesc) END;

    D = POINTER TO DDesc;
    DDesc = RECORD (BDesc) END;

    E = POINTER TO EDesc;
    EDesc = RECORD (CDesc) END;

    F = POINTER TO FDesc;
    FDesc = RECORD (CDesc) END;

  PROCEDURE (x: A) Print;
  BEGIN Out.Char("A")
  END Print;

```

```

PROCEDURE (x: C) Print;
BEGIN x.Print; Out.Char("C")
END Print;

```

```

PROCEDURE (x: F) Print;
BEGIN Out.Char("F")
END Print;

```

```

PROCEDURE (x: D) Print;
BEGIN Out.Char("D")
END Print;

```

```

PROCEDURE Start*;
  VAR a: A; b: B; c: C; d: D; e: E; f: F;
  BEGIN
    NEW(a); NEW(b); NEW(c); NEW(d); NEW(e); NEW(f);
    a.Print; b.Print; c.Print; d.Print; e.Print; f.Print; Out.Ln
  END Start;

```

```

END KlausurAufgabe4.

```

```

KlausurAufgabe4.Start

```