Maple-Test zur LVA 'Computermathematik'

**** MIT LÖSUNGEN (Codes, Erläuterungen) ****

** GRUPPE GELB **

7. Mai 2009

NAME, Vorname: .....

Matr.Nr.: .....

- **8 Aufgaben**
- **30 Punkte + ggf. 4 Bonuspunkte 'extra' (Aufgabe 8)**
- **Die Verwendung schriftlicher Unterlagen ist nicht gestattet.**
- **Unten zu formulierende Prozeduren:**
  generell ohne spezielle Parameterüberprüfung.
- **Bitte verwenden Sie jeweils nur das eine Blatt für ihre**
  Lösung. Keine Extra-Blätter beilegen.

==========

===== AUFGABE 1 (3 Punkte):

Was passiert bei nachfolgendem Aufruf f('n')?

```
> f:=proc(k)
  local j,a;
  a:=0;
  for j from k to 1 by -1 do
    a:=a*j
  end do
  end proc;
```

f := **proc**(*k*) **local** *j*, *a*; *a* := 0; **for** *j* **from** *k* **by** -1 **to** 1 **do** *a* := *a*∗*j* **end do end proc**

```
> f('n');
Error, (in f) initial value in for loop must be numeric or character
```

Lösung:

-- Schleife nicht ausführbar (kein numerischer Wert für obere Grenze übergeben)

===== AUFGABE 2 (3 Punkte):

(a) Erklären Sie jeden einzelnen Befehl.
(b) Welchen Typ hat das Objekt Z ?
(c) Geben Sie an, wie die Ausgabe aussieht.

```
> n:=6;
```

$$n := 6$$

```
> Z:=[seq(i,i=1..n)];
```

$$Z := [1, 2, 3, 4, 5, 6]$$

```
> for j from 1 to nops(Z)-1 do
      Z[j]:=Z[nops(Z)-j]
  end do;
```

$$Z_1 := 5$$
$$Z_2 := 4$$
$$Z_3 := 3$$
$$Z_4 := 4$$
$$Z_5 := 5$$

Lösung (Hinweise):

-- seq(...) konstruiert expression sequence, [...] ist die daraus entstehende Liste

-- nops (...) liefert Anzahl der Elemente der Liste Z

-- sinmple Schleife

===== AUFGABE 3 (4 Punkte):

Schreiben Sie eine Prozedur q(l), die eine gegebene Liste l von integers analysiert, die Anzahl der durch 3 teilbaren geraden Zahlen in l feststellt und diese Anzahl gemeinsam mit dem Produkt dieser Zahlen in einer Liste (2 Elemente) zurückgibt.

Hinweis: - is(n,even) testet auf 'gerade' (--> true/false)
- m mod n liefert m modulo n

Lösung: (Code der Prozedur q):

```
> q := proc(l::list)
  local il, Nl, number, counter, prod;
  Nl := nops(l);
  counter := 0;
  prod := 1;
  for il from 1 to Nl do
    number := l[il];
    if (number mod 6 = 0) then # Test auf Teilbarkeit durch 6
      counter := counter + 1;
      prod := prod * number
    end if
  end do;
  return [counter,prod]
  end proc:
```

Beispiel:

```
> q([2,3,7,8,12,18,19]);
```

[2, 216]

===== AUFGABE 4 (4 Punkte):

Schreiben Sie eine Prozedur poleval(c,t), die
-- die Elemente c[0], c[1], ... eines gegebenen Arrays c
(n+1 Einträge, Startindex 0) als Koeffizienten eines
Polynoms interpretiert,
-- t als Auswertungsstelle interpretiert
-- und den Wert des Polynoms an der Stelle t
(also: summe von j=0...n: c[j]*t^j),
mittels des Horner-Schemas berechnet und zurückgibt.

Hinweis: -- Horner-Schema: Auswertung gemäß
c[0] + t * (c[1] + t * (c[2] + ...))
Überlegen Sie die Organisation der Schleife!
-- ArrayNumElems(...) liefert Anzahl der Elemente eines Arrays

Lösung (Code der Prozedur poleval):

```
> poleval := proc(c,t)
  local j, n, polval;
  n := ArrayNumElems(c)-1;
  polval := c[n];
  for j from n-1 to 0 by -1 do
    polval := t*polval + c[j]
  end do;
  return polval;
  end proc:
```

Beispiel:

```
> 1+2*x+4*x^3;
```

$$1 + 2\,x + 4\,x^3$$

```
> poleval(Array(0..3,[1,2,0,4]),x);
```

$$x\,(4\,x^2 + 2) + 1$$

```
> expand(%);
```

$$1 + 2\,x + 4\,x^3$$

```
> poleval(Array(0..3,[1,2,0,4]),sqrt(2));
```

$$10\,\sqrt{2} + 1$$

===== AUFGABE 5 (4 Punkte):

Schreiben Sie eine Prozedur listdiff(x,y), die zu zwei gegebenen Listen x, y Folgendes zurückgibt:

-- x, y haben verschiedene Länge: FAIL

-- x, y haben gleiche Länge: [[*b*,*b*,...*b*],m]
wobei b = true falls die jeweiligen Listenelemente übereinstimmen,
b = false sonst,
und m = Anzahl der Übereinstimmungen

Lösung (Code der Prozedur listdiff):

```
> listdiff := proc(x::list,y::list)
  local counter, i, Lout;
  if (nops(x)<>nops(y)) then
    return FAIL
  else
    counter := 0;
    Lout := [seq(false,iL=1..nops(x))]; # Initialisierung
    for i from 1 to nops(x) do
      if (x[i]=y[i]) then
         Lout[i] := true;
         counter := counter + 1
      end if
    end do;
    return [counter,Lout]
  end if
  end proc:
```

Beispiel:

```
> listdiff([a,b],[a,b,c]);
```

FAIL

```
> listdiff([1,2,3,4,5],[5,4,3,2,1]);
```

[1, [*false*, *false*, *true*, *false*, *false*]]

===== AUFGABE 6 (4 Punkte):

Schreiben Sie eine Prozedur gsolve(xstart,g::procedure,k,delta),
die für eine gegebene skalare reelle Funktion g(x)
k Schritte der Newton-Iteration durchführt (Nullstellensuche).
Die Iterationsvorschrift lautet: x[0]:=xstart, und

x[j] := x[j-1] - g(x[j-1]) / g'(x[j-1]), j=1,2,...,k

Zurückzugeben sind alle Iterierten x[0],...x[k] in einem Array x.
Rechnung in Gleitpunktarithmetik (evalf(...)).

Der Parameter delta soll festlegen, wann
die Iteration vorzeitig abzubrechen ist, d.h.:

Falls |x[j]-x[j-1]| < delta: Iteration wird beendet,
und es wird nur x[0],...x[j] zurückgegeben..

Lösung (Code der Prozedur gsolve):

```
> gsolve := proc(xstart,g::procedure,k,delta)
  local j, x;
  x := Array(0..k);
  x[0] := xstart;
  for j from 1 to k do
    x[j] := evalf(x[j-1]-g(x[j-1])/D(g)(x[j-1]));
    if (abs(x[j]-x[j-1]) < delta) then
      return x[0..j]
    end if;
  end do;
  return x[0..k-1];
  end proc:
```

Beispiel (Quadratwurzel aus 2):

```
> gsolve(2,x->x^2-2,5,1e-2);
```

Array(0 .. 3, {(0) = 2, (1) = 1.500000000000000000000000000,
(2) = 1.416666666666666666666666667, (3) = 1.414215686274509803921568627 45},
datatype = *anything*, *storage* = *rectangular*, *order* = *Fortran_order*)

```
> gsolve(2,x->x^2-2,5,1e-10);
```

Array(0 .. 5, {(0) = 2, (1) = 1.500000000000000000000000000,
(2) = 1.416666666666666666666666667, (3) = 1.414215686274509803921568627 45,
(4) = 1.414213562374689910626295578 89, (5) = 1.414213562373095048801689623 50},
datatype = *anything*, *storage* = *rectangular*, *order* = *Fortran_order*)

```
> gsolve(0,x->x^2-2,5,1e-10);
Error, (in gsolve) numeric exception: division by zero
```

===== AUFGABE 7 (4 Punkte):

Beschreiben und erklären Sie die Syntax der wichtigsten Steuerkonstrukte in Maple (Verzweigungen, Schleifen, ...).

Lösung:

siehe VO

===== AUFGABE 8 (4 Punkte + 4 Bonuspunkte bei perfekter Lösung!)

Was tut die folgende Prozedur f ?
Erklären Sie genau, was f bei korrektem Aufruf zurückliefert.
Erklären Sie auch die error-Meldung.

```
> f:=proc(p::procedure,K::posint)
  local c,d,k,t;
  c:=Array(0..K+1);
  d:=Array(0..K+1);
  for k from 0 to K+1 do
     d[k]:=`if`(k=0,p(t),diff(p(t),t$k));
     c[k]:=subs(t=0,d[k])/k!;
     if (simplify(d[k]=0)) then
        return k-1,c[0..k-1]
     end if
  end do;
  error ("K too small? You may try to increase K.");
  end proc:
```

Lösung (Hinweis):

- f stellt fest, ob p ein Polynom vom (Maximal)Grad N ist.
- Grundlage: Test der höheren Ableitungen auf =0 (identisch 0 !);
  Berechnung der Koeffizienten nach dem Taylor'schen Satz.

"error": Falls N nicht groß genug gewählt, unentscheidbar. (Höherer Grad? kein Polynom?)

Beispiel:

```
> p:=x->1+x+5*x^5;
```

$$p := x \to 1 + x + 5\,x^5$$

```
> f(p,4);
Error, (in f) K too small? You may try to increase K.
```

```
> f(p,5);
```

$$5, \mathrm{Array}(0 .. 5, \{(0) = 1, (1) = 1, (5) = 5\}, \mathit{datatype} = \mathit{anything}, \mathit{storage} = \mathit{rectangular}, \mathit{order} = \mathit{Fortran\_order})$$

Achtung: Das ist nicht 100% verlässlich.. Beispiel:

```
> p := x->cos(3*arccos(x));
```

$$p := x \to \cos(3 \arccos(x))$$

```
> expand(p(x));
```

$$4\,x^3 - 3\,x$$

```
> f(p,3);
Error, (in f) K too small? You may try to increase K.
```

... wird von obigem Code nicht erkannt. (Das wäre eine weitere Aufgabe, deutlich subtiler.)