BuK 2000 – Lösungen Übungen

Dies sind private Lösungen, KEINE Musterlösungen, und somit nicht unbedingt korrekt!
Trotzdem helfen Sie vielleicht.
Wenn jemand die Musterlösungen hat – her damit! http://s-inf.de

BuK – Übung Nr. 2

Klaus Ridder 225871, Matthias Hensler 219426

Aufgabe 3:

Gehe bei jeder Ausgabe alle bisher ausgegebenen Wörter durch, ob dieses bereits vorgekommen ist. Wenn ja, mache keine Ausgabe, sonst gebe aus.

Aufgabe 4:

à:
wir gehen alle Wörter W in der kanonischen Reihenfolge durch, und überprüfen jeweils, ob die gegebene Funktion mit diesem Wort als Eingabe 0 oder 1 liefert. (Sie liefert nach Voraussetzung auf jeden Fall 0 oder 1, da W „entscheidbar“ ist.)
Liefert die Funktion nun 1, geben wir das Wort aus, sonst nicht.
Alle Wörter der Wortmenge W sind dann in kanonischer Reihenfolge aufgezählt.
qed.

ß:
Jetzt haben wir als Eingabe ein Wort x. Die Ausgabe soll 0 oder 1 sein, je nach dem, ob x in W enthanlten ist oder nicht.
Wir starten nun unseren vorhandenen Aufzählungsalgorithmus, der uns alle Worte W in kanonischer Reihenfolge ausgibt.
Wir schauen uns jetzt jedes Wort an, den uns der Algorithmus liefert. Entspricht dieses Wort unserem eingegegbenen Wort x, so terminiere mit Ausgabe 1. Ist die Position dieses Wort in kanonischer Reihenfolge größer als unser eingegebenes x, so terminiere mit Ausgabe 0. Ansonsten betrachte das nächste Wort, dass uns der Algorithmus liefert.
Dieser Algorithmus terminiert immer, da immer x oder ein Wort gefunden wird, dass größer als x ist, da der Algorithmus die Worte in kanonischer Reihenfolge sortiert ausgibt.
qed.

Aufgabe 5:

Dieses Problem ist nicht entscheidbar ist, da keine Information vorhanden ist, ob das Programm überhaupt terminiert.

BuK – Übung Nr. 3

Klaus Ridder 225871, Matthias Hensler 219426

Aufgabe 6

Turingmaschine, die eine binäre Eingabe invertiert.
Voraussetzung: Eingabe ist an einem Stück, und an beiden Seiten durch # begrenzt.
Beispiel-Eingabe: #001#

Format der Turing-Maschine:
Startzustand(q0), LeseZeichen, SchreibZeichen, Kopfbewegung, NeuerZustand

Voraussetzung: unsere Turing-Maschine steht auf der # vor der Zahlenkette.
Turing-Maschine:

q0, #, #, R, q1
q1, 0, 1, R, q1
q1, 1, 0, R, q1
q1, #, #, L, q2
q2, 0, 0, L, q2
q2, 1, 1, L, q2
q2, #, #, N, qs.

Diese Turing-Maschine bewegt den Kopf von der linken # nach rechts bis zur rechten #, und ersetzt dabei jede0 durch eine 1. Dann geht sie – ohne Veränderungen vorzunehmen – zur ersten # zurück.

Aufgabe 7:

Man muss jede Aktion „N“ durch eine Bewegung nach oBdA rechts und dann zurück nach links ersetzen.
Die „N“ – Operation ist dann nur so zu verändern, dass statt „N“ „R“ ausgeführt wird.
Die neu eingefügte Operation muss nun dasselbe Zeichen wieder schreiben, das vorher an dieser Position stand (also * in abkürzender Schreibweise), und dann nach links gehen. Fertig.

Aufgabe 8:

Semi – Entscheidbar = Es gibt einen Algorithmus, der feststellen kann, ob das gesuchte Wort in der Wotmenge enthalten ist. Ist es drin, gibt er 1 aus, ansonsten terminiert er nicht.

aufzählbar = man kann die einzelnen Wörter in einer eindeutigen Reihenfolge hintereinander packen.

aufzählbar à semi-entscheidbar:
starte den Aufzählungsalgorithmzus. Sobald das gesuchte Wort gefunden ist, gebe 1 aus.
ENDLICHE MENGE: Wenn alle durch sind, und das Wort nicht gefunden wurde, gehe in eine Endlosschleife,
UNENDLICHE MENGE: Suche einfach unendlich weiter.

semi-entscheidbar à aufzählbar:
hier müssen wir wieder die „Diagonal-Methode“ (oder wie die auch hies) anwenden:

1.) gehe 1 Schritt beim 1. Wort
2.) gehe einen weiteren Schritt beim 1. und beim 2. Wort
3.) gehe einen weiteren Schritt beim 1., 2. und 3. Wort
4.) gehe einen weiteren Schritt beim 1., 2., 3. und 4. Wort
5.) gehe einen weiteren Schritt beim 1., 2., 3., 4. und 5. Wort
6.) etc.

sobald wir für ein Wort die Antwort bekommen, geben wir es aus. **à** fertig.

BuK – Übung Nr. 4

Klaus Ridder 225871, Matthias Hensler 219426

Aufgabe 9

WHILE – Programm, dass GGT berechnet:

```
X:=Eingabe_1;  // z.B. 729
Y:=Eingabe_2;  // z.B. 153
Rest:=1;  // Rest ungleich 0, zum 1. Betreten der WHILE-Schleife

WHILE Rest<>0 DO {
        // suche größere der beiden Eingaben
        IF Y>X THEN { Z:=Y; Y:=X; X:=Z}  // größere Variable nach X, z.B. 729
        Counter:=0
        WHILE Y*(Counter+1)<X DO Counter:=Counter+1; // Counter=4
        Rest:=X-(Y*Counter);
        X:=Y;
        Y:=Rest;
}
```

Das Ergebnis steht nun in X, z.B. „RETURN(X);“
Es wird vorausgesetzt, dass es immer einen GGT gibt!

BEISPIEL:

729 = **153** * 4 + 117
153 = **117** * 1 + 36
117 = **36** * 3 + 9
36 = **9** * 4 **à** ggT (729, 153) = 9

Aufgabe 10

f(1)=1
f(2)=1+2
f(3)=1+2+3
f(n)=1+...+n = (n+(n*n))/2

LOOP-Berechenbar: wie WHILE-Berechenbar, aber ohne WHILE.

(a):

```
Ergebnis:=0;
i:=1;
LOOP n {
        Ergebnis.=Ergebnis+i;
        i:=i+1;
}
```

(b):

```
n:=Eingabevariable;
X:=1;
LOOP n {
        IF ((X1+1)*(X+1))<n THEN X:=X+1;
}
```

Aufgabe 11: ?

BuK – Übung Nr. 5

Klaus Ridder 225871, Matthias Hensler 219426

Aufgabe 12

a) Berechnung der Fibonacci – Funktion:

```
n:=Erster_Paramter_also_unser_n_aus_der_Aufgabe;

IF (n=0 OR n=1) THEN Fib:=1;
ELSE {
        Fib_1vorher:=1;
        Fib_2vorher:=1;
        m:=2;
        WHILE m<=n DO {
                Fib:=Fib_1vorher+Fib_2vorher;
                Fib_2vorher:=Fib_1vorher;
                Fib_1vorher:=Fib;
                m:=m+1;
        }
}
```

Das Ergebnis steht dann in **Fib**.

PROBE: n=4:

Schritt	Fib	Fib_1vorher	Fib_2vorher	m
1	-	1	1	2
2	2	2	1	3
3	3	3	2	4
4	5	5	3	5
5	5			

Fib(4)=Fib(3)+Fib(2)=Fib(2)+Fib(1)+Fib(1)+Fib(0)=Fib(1)+Fib(0) +Fib(1)+Fib(1)+Fib(0)=5 correct.

b) f(n) = 2^(2^n)

Vorgehensweise: $2^n = 2*2*2*2*2*….*2$, n-mal.

```
Wert:=1;  Loop n {Wert:=Wert*2;};
m :=Wert;
Wert:=1;  Loop m {Wert:=Wert*2;};
```

In **Wert** steht nun das Ergebnis.

Aufgabe 13:

Emulation von IF...THEN...ESEL durch WHILE:

IF Bedingung THEN Prog1 ELSE Prog2; **à**

```
Temp:=1;
WHILE Bedingung AND Temp DO {Prog1; Temp:=0}
WHILE (Temp=1) DO {Prog2; Temp:=0};
```

Aufgabe 14:

```
LOOP y {Funktion};
```

BuK – Übung Nr. 6

Klaus Ridder 225871, Matthias Hensler 219426

Aufgabe 18

(P1)
trivial lösbar: man lasse die Turingmaschine einfach n Schritte laufen und schaue, ob sie stoppt.
(P2)
das ist trivialerweise nicht lösbar: es ist schwieriger als das Wortproblem:
man muss ihn erst mal n Schritte laufen lassen. Stoppt er nach weniger n Schritten, so könnten wir NEIN ausgeben. Ansonsten haben wir das Wortproblem.

WP<H2.
f : E_{WP} **à** E_{P2} mit folgender Transformation:
M : w **à** Stopp ⇔ M' : w **à** Stopp mit Schrittzahl größer n:

wir arbeiten zunächst m Schritte wie m, und geben nein aus, falls wir dabei stoppen. Sonst arbeiren wir weiter wie m.

Aufgabe 19

$H_{111} \leq H$:
Finde also: f : E_{H111} **à** E_H mit M : 111 **à** Stopp. ⇔ M' : ε **à** Stopp.
Konstruiere M' wie folgt: M' schreibt zunächst 111 auf das leere Band, geht dann auf das erste Feld zurück und arbeitet dann wie M.
Somit ist klar, dass f : (M,111) **à** M' berechenbar ist. Nach Konstruktion von M' gilt dann auch, dass M:111 **à** Stopp ⇔ M' : ε **à** Stopp , d.h. (M, 111) ∈ H_{111} gdw. M' ∈ H.

$H \leq H_{111}$:
Finde also: f: E_H **à** E_{H111} mit M:ε **à** Stopp ⇔ M':111 **à** Stopp .
Konstruiere M' wie folgt: Lösche die Eingabe 111 und arbeite einfach wie M.
Somit ist klar, dass f : (M,ε) **à** M' berechenbar ist. Nach Konstruktion von M' gilt dann auch, dass M:ε **à** Stopp ⇔ M' : 111 **à** Stopp , d.h. (M, ε) ∈ H gdw. M' ∈ H_{111}.

Aufgabe 20

Es gibt maximal c = Anzahl_Zustände * Anzahl_Zeichen Konfigurationen.
wir gehen nun maximal c+1 Schritte. Wenn das Turingprogramm bis dahin nicht gestopt hat, bedeutet dies, dass mindestens eine Konfiguration doppelt besucht wurde, und wir somit mindestens LOOP – Schlife haben. Begründung: Wenn eine TM zweimal die gleiche Konfiguration „besucht", ist es sicher, dass es nie terminiert, da dies einer LOOP-Schleife gleichkommt. (Ddie TM kann ja nicht „zählen", wie oft sie dort reingeht, und auch keine Bedingungen checken). q.e.d.

BuK – Übung Nr. 8

Klaus Ridder 225871, Matthias Hensler 219426

Aufgabe 22

(a)	(b)	(c)	(d)
1 2 1 bbbabbbbb bbbabbbbb	unlösbar. Grund: Egal ob man mit 1, 2 oder 3 anfängt, die Folgen unterscheiden sich bereits.	1 1 2 3 1 1 aaaabbaaaaa aaaabbaaaaa	Es gibt keine Lösung: Sowohl die Pfade 2 1 * als auch 2 3 * führen zu einem Widerspruch.

Aufgabe 23

da wir nur endlich viele Steintypen haben, kommen wir entweder an einer stelle nicht weiter, oder wir wiederholen uns („Periode“).

Folgender Algorithmus löst das Problem:
Lege d0 hin.
Starte (wie in einer Baumstruktur) nun für jeden möglichen anlegbaren Stein eine Rekursionsinstanz. (wichtig: merke dir alle benutzen Steine auf dem Pfad!)

Lege also den ersten Stein an.
Nun gibt es 2 Möglichkeiten:
Möglichkeit 1: einer der angelegten Steine hat auf der rechten Seite das gleiche Symbol wie einer der bisher auf diesem Pfad angelegten Steine (wir haben sie ja zwischengespeichert), so ergibt sich eine „Periode“, und die Zeile ist unendlich parkettierbar.
Möglichkeit 2: wir stoßen in allen unseren Rekursionspfaden auf eine unlösbare Stelle. Dann gibt es keine Parkettierung.

Wegen der Endlichkeit der Steinarten tritt immer in endlicher Zeit eine der beiden Möglichkeiten ein.

BEISPIEL: Steine 1-2, 2-4, 2-5, 5-2.

1-**2** 2-4 **à** dead end
2-5 5-**2** **à** Schleife.

Aufgabe 23'

Wir haben alle Wörter, die nur a enthalten, also a, aa, aaa, aaaa, ect.

BEISPIEL:
(a,aa) (aaa,aaaa) **à** keine Lösung
(a,aaa) (aa, aaaa) **à** Lösung 1,2

LÖSUNG:
Für die Lösbarkeit muß es sowohl einen Index geben, in dem Wortlänge2 ≥ Wortlänge1 und einen Index Wortlänge1 ≥ Wortlänge 2. (dies kann bei Gleichheit derselbe Index sein).
Sonst lösbar mit folgendem Algorithmus:
Haben wir einen gleichlangen Index, nehme diesen oBdA ein mal **à** fertig.
Ansonsten nehme 2 Indizes, einmal wo der erste, einmal wo der zweite kürzer ist.
Betrachte nun die Differenz der Längen zwischen den Indizes (Diff1 und Diff2).
Schreibe nun den ersten Diff2-mal hin, und den zweiten Diff1-mal **à** fertig.

BEISPIEL: (aaa, aaaaa) (aaaa, a) Diff1 = 1, Diff2 = 4.
Schreibe also Index1 4-mal und Index2 1-mal hin:
aaa aaa aaa aaa aaaaa
aaaa aaaa aaaa aaaa a. Damit ist das Problem entscheidbar.