# Übungsblock 3

Lösungen 2.Übungsblatt:

Aufgabe 1: a) partiell, da 5/x für x=0 nicht definiert;
Definitionsbereich: alle x aus IR außer x=0

b) total

c) partiell; Definitionsbereich: alle negativen geraden ganzen Zahlen ≤ 0

Aufgabe 2: siehe aufg2_2.java

Aufgabe 3: umgangsprachlich:

solange 0: nach rechts gehen

wenn 1: teste, ob daneben zweimal eine 1 und dann eine 0,

wenn ja: gehe 3 Schritte zur linken 1 zurück

wenn nein: laufe nach rechts bis zur 0 und starte neu

berücksichtige: Einsen von Null umschlossen → anfänglich bis zur 1. Null laufen!

# Übungsblock 3

Aufgabe 3:

| akt. Zust. | gel.Zeichen | | schreibe | neuer Z. | Kopfbew. | |
|---|---|---|---|---|---|---|
| s0 | 0 | → | 0 | s1 | r | |
| s0 | 1 | → | 1 | s0 | r | |
| s1 | 0 | → | 0 | s1 | r | |
| s1 | 1 | → | 1 | s2 | r | |
| s2 | 0 | → | 0 | s1 | r | |
| s2 | 1 | → | 1 | s3 | r | |
| s3 | 0 | → | 0 | s1 | r | |
| s3 | 1 | → | 1 | s4 | r | |
| s4 | 0 | → | 0 | s5 | l | |
| s4 | 1 | → | 1 | s0 | r | |
| s5 | 0 | → | 0 | s0 | r | // muss nicht |
| s5 | 1 | → | 1 | s6 | l | |
| s6 | 0 | → | 0 | s0 | r | // muss nicht |
| s6 | 1 | → | 1 | s7 | l | |

Startzustand: s0, Endzustand={s7}, Zustandsmenge={s0,s1,s2,s3,s4,s5,s6,s7}
Bandalphabet={0,1}

# Übungsblock 3

Aufgabe 4: Java-Programm realisiert sog. Heron-Verfahren zur Berechnung der Quadratwurzel aus gegebenem Parameter.

Iterative Annäherung an Seitenlänge eines Quadrats mit gegebenem Parameter als Flächeninhalt: ausgehend von Rechteck mit gleichem Flächeninhalt werden die Seiten a und b über a'=(a+b)/2, b'=(Parameter/a') nach und nach dem Wurzelwert angenähert.

Programm erwartet mind. einen Kommandozeilenparameter, sonst Fehlermeldung und Programmende (21-23).

Erster Kommandozeilenparameter wird in Integer gewandelt. Ist der Parameter nicht vom Typ int, so Fehlerabfang durch try-catch mit Programmende (24,34-36). Ist Parameter negativ: Fehlermeldung und Programmende (25-29).

Funktion teiler (30, 4-8) ermittelt von 2 ausgehend kleinsten Teiler des Parameters, indem fortlaufend durch 2,3,4,… geteilt wird, bis Division keinen Rest liefert (6). Dann ist kleinsterTeiler(<> 1) erreicht und wird zurückgegeben (7).

Funktion erg (32, 10-17) erhält Int-Parameter, kleinsten Teiler und (Parameter/kl.Teiler) als Parameter und ermittelt gemäß obigem Iterationsverfahrens (13,14) die Wurzel mit einer Genauigkeit von 0.0001. D.h., die Berechnung bricht ab, sobald die Seitenlängen x und y sich nur noch um max. 0.0001 unterscheiden (12).

Das Programm liefert die Ausgabe:"Die gesuchte Zahl lautet *[wurzelwert]*" (32).

Ein try-catch-Block fängt alle weiteren Fehler ab (24,34-36).

# Übungsblock 3

## Handhabung clisp:

1. Starten mit

   clisp

   erzeugt eine Read-Eval-Print-Loop, in der die Funktionsdeklarationen direkt eingegeben werden können und von clisp direkt ausgewertet werden.
   (Hinweis: ` ist shift-#)

2. Verlassen der Read-Evel-Print-Loop mit

   (quit)

3. Alternativ: Funktionsdefinitionen in dateiname.lisp eingeben, clisp starten mit

   clisp –i dateiname.lisp

   Inhalt der Datei wird geladen (egal ob source-Code oder compilierter Code) und anschließend die Read-Eval-Print-Loop gestartet. Funktionen aus dateiname.lisp sind dann verfügbar. Angabe von mehreren –i-Optionen ist erlaubt.

4. Optionen beim Starten:

| | | |
|---|---|---|
| a) | clisp –L german | startet clisp mit deutschen Kommentaren |
| b) | clisp –c dateiname.lisp | compiliert dateiname.lisp; Bytecodedatei erhält Endung .fas |
| c) | clisp –c dateiname.lisp –o name | wie b., nur Bytecodedatei heißt name |
| d) | clisp –repl | startet Read-Eval-Print-Loop nach –c-Option |
| e) | clisp dateiname.lisp | clisp lädt Ausdrücke der Datei und führt sie aus (nur die Ausgaben auf standard-out sind sichtbar) |