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: umgangssprachlich:

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 berechnet zu zwei Integerzahlen die XOR-Verknüpfung ihrer Binärdarstellung und gibt diese wieder als Integerzahl aus.**

Programm erwartet mindestens zwei Kommandozeilenparameter, sonst Fehlermeldung und Programmende (28-29).

Die ersten beiden Kommandozeilenparameter werden in Integer gewandelt. Sind die Parameter nicht vom Typ int, so Fehlerabfang durch try-catch mit Programmende (27, 36-37). Aufruf der Funktion wasBinIch, die als Parameter die Binärdarstellung der beiden Programmparameter bekommt (33). Der try-catch-Block fängt alle weiteren Fehler ab (27,36-37).

wasBinIch (12-23) testet als erstes, ob übergebene Binärdarstellungen unterschiedlich lang sind. Wenn ja, wird für die kürzere Darstellung die Funktion verl aufgerufen, die außerdem die längere Länge als Parameter bekommt. (15,16)

verl (4-11) erhält einen String und eine Länge und erzeugt einen neuen String, dessen ersten Zeichen 0 sind (Anzahl: Differenz aus alter Stringlänge und übergebener Länge) (7,8). Die restlichen Zeichen des neuen Strings werden vom übergebenen String übernommen (9).

Nach Aufruf von verl sind in wasBinIch beide Binärdarstellungen gleich lang. Nun wird Charakter-Array mit gleicher Länge erzeugt (17). Die Felder werden auf 0 gesetzt (19) und nur dann in 1 geändert, wenn in den Binärdarstellungen an der jeweils betrachteten Stelle zwei unterschiedliche Bits stehen (20). Damit entspricht das Ergebnis-Array der XOR-Verknüpfung der Binärstrings.

Das Array wird als Binär-String zurückgegeben (22), in die zugehörige Integerzahl umgewandelt und mit dem Text „*Die gesuchte Zahl lautet:*“ ausgegeben (34).

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)