Aufgabe 1 (Lex und Yacc) ***19 Punkte***

Ziel dieser Aufgabe ist es, mit Hilfe von Lex und Yacc römische Zahlen in Dezimalzahlen umzuwandeln. Bekanntermaßen stehen im römischen Zahlensystem die Ziffern (bzw. Buchstaben) I für 1, V für 5, X für 10, L für 50, C für 100, D für 500 und M für 1000 zur Verfügung. Die Zeichen können unter Beachtung bestimmter Regeln nebeneinander geschrieben werden. Die Regeln lauten wie folgt:

R1: Die Zeichen I, X, C und M dürfen wiederholt nebeneinander stehen, die Zeichen V, L und D jedoch nicht.

R2: Stehen gleiche Zeichen nebeneinander, so werden sie addiert (z.B. XXX = 30 = 10 +10+10). Jedoch dürfen die Zeichen I, X und C nicht mehr als dreimal nebeneinander auftreten. Das Zeichen M kann beliebig oft nebeneinander angegeben werden.

R3: Die Zeichen V, L und D dürfen in einer Zahl nur einmal vorkommen.

R4: Steht ein Zeichen für eine kleinere Einheit rechts neben einem Zeichen für eine größere Einheit, dann wird die kleinere Einheit auf die größere Einheit addiert (z.B. DCCLVI = 756 = 500+100+100+50+5+1).

R5: Steht ein Zeichen für eine kleinere Einheit links neben einem Zeichen für eine größere Einheit, dann wird die kleinere Einheit von der größeren Einheit subtrahiert (z.B. XXIX = 29 = 10+10+10-1).

R6: Es dürfen nicht zwei oder mehr kleinere Einheiten von einer rechts stehenden größeren Einheit abgezogen werden.

Die Darstellung einer Dezimalzahl durch eine römische Zahl ist nicht eindeutig. Es gilt z.B. MIM = MCMIC = MCMXCIX = 1999.

(a) Schreiben Sie ein Lex-Programm, das eine Folge von durch Returns getrennten römischen Zahlen als Eingabe entgegen nimmt, die Eingabe lexikalisch analysiert und Vorkehrungen für die Syntaxanalyse mit Yacc trifft. Beachten Sie, daß auch eine Kleinschreibung der römischen Zahlen erlaubt ist. *5 Punkte*

(b) Schreiben Sie ein Yacc-Programm, das die Eingabe syntaktisch analysiert, bei korrekter Syntax die entsprechende Dezimalzahl berechnet, bei Verletzung der Regel R3 anstelle der Berechnung eine Warnung ausgibt und ansonsten einen Syntaxfehler meldet. *14 Punkte*

Aufgabe 2 **(LL(1)-Grammatiken)** ***12 Punkte***

Gegeben sei die folgende Grammatik $G = (N, \Sigma, P, S)$ mit

$N = \{S, E, L\}$,

$\Sigma = \{\textbf{id}, \textbf{print}, \textbf{num}, ;, ,, (,), :=, +\}$ und

$P = \{ S \to S ; S \mid \textbf{id} := E \mid \textbf{print}\ (L),$

$E \to \textbf{id} \mid \textbf{num} \mid E + E,$

$L \to E \mid L, E \}.$

Überführen Sie G in eine äquivalente LL(1)-Grammatik G'.

1. $S \to id := E S' \mid print (L) S'$

$S' \to ; S S' \mid \varepsilon$

2. $E \to id E' \mid num E'$

$E' \to + E E' \mid \varepsilon$

3 $L \to E L'$

$L' \to , E L' \mid \varepsilon$

Aufgabe 3 **(FIRST- und FOLLOW-Mengen)** ***24 Punkte***

Gegeben sei die folgende LL(1)-Grammatik $G = (N, \Sigma, P, S)$ mit

$N = \{S, A, B, C, D, E, F, G\}$,

$\Sigma = \{\textbf{a}, \textbf{b}, \textbf{c}, \textbf{d}, (,), ;, :\}$ und

$P = \{$

(1) $S \to \textbf{c}\,\textbf{a}\,A\,;$

(2) $A \to (\,B\,)$

(3) $A \to \varepsilon$

(4) $B \to D\,E$

(5) $B \to C$

(6) $C \to \textbf{a} : \textbf{b}\,F$

(7) $F \to ;\,\textbf{a} : \textbf{b}\,F$

(8) $F \to \varepsilon$

(9) $D \to \textbf{d}\,\textbf{a} : \textbf{b}\,;\,G$

(10) $G \to D$

(11) $G \to \varepsilon$

(12) $E \to C$

(13) $E \to \varepsilon$

$\}$

(a) Berechnen Sie die *initialen Steuermengen*. Erstellen Sie zunächst gemäß dem im Kurs angegebenen Verfahren einen Graphen, der die Reihenfolge zur Berechnung der FIRST-Mengen angibt. *10 Punkte*

(b) Bestimmen Sie die Steuermengen. Berechnen Sie dazu die FOLLOW-Mengen mit Algorithmus 3.15 und geben Sie auch den Graphen an, der aus der Anwendung dieses Algorithmus resultiert. *10 Punkte*

(c) Stellen Sie die Analysetafel auf. *4 Punkte*

Aufgabe 4 (Übersetzung funktionaler Sprachen) *20 Punkte*

(a) Geben Sie die allgemeinsten Typen der folgenden ML-Funktionen an. Begründen Sie Ihre Antworten. *(6 Punkte)*

(i) **fun** $f(x, y) = x+1$

(ii) **fun** $g(x, y)$ = **if** $x=y$ **then** 0 **else** 1

(iii) **fun** $h(x, y)$ = **if** $x>3$ **then** $h(y, x)$ **else** $h(x, y)$

(b) Werten Sie den folgenden Mini-ML-Ausdruck mit dem im Abschnitt 7.3 angegebenen Interpreter in der Umgebung U aus, wobei U die Bindung $(z, 7)$ enthalte. Geben Sie alle Zwischenschritte an, und nennen Sie jeweils die Regeln, die angewendet werden. *(7 Punkte)*

(**fn** $i \Rightarrow i*2$) z

(c) Der folgende Mini-ML-Ausdruck realisiert eine Funktion zur Berechnung des Maximums zweier Zahlen. Übersetzen Sie diesen Ausdruck in SECD-Code. *(7 Punkte)*

fn $x \Rightarrow$ **fn** $y \Rightarrow$ **if** $x>y$ **then** x **else** y

Aufgabe 5 (Flußgraphen) *15 Punkte*

Gegeben sei das folgende Programmstück im 3-Adreß-Code:

```
(1) if x<3 then goto 6
(2) x := x-1
(3) y := y-1
(4) if y>7 then goto 8
(5) y := x
(6) x := y+2
(7) goto 3
(8) x := y-1
```

(a) Bestimmen Sie die Basisblöcke zu diesem Programmstück. *(3 Punkte)*

(b) Geben Sie den Flußgraphen an. *(3 Punkte)*

(c) Ermitteln Sie die Mengen $gen(B_i)$, $kill(B_i)$, $in(B_i)$ und $out(B_i)$ $(1 \leq i \leq 6)$ zur Bestimmung der UD-Verkettung. *(9 Punkte)*

Aufgabe 6 (Codeerzeugung) ***10 Punkte***

Erzeugen Sie Code für den folgenden Basisblock unter der Annahme, daß zwei Register `s` und `T` zur Verfügung stehen. Gehen Sie weiterhin davon aus, daß am Ende des Blocks lediglich die Variable `z` zu speichern ist.

```
T1 := a-b
T2 := c*d
T3 := T2+x
T4 := T3-T1
T5 := T4+y
T6 := T2*T4
z  := T6+T5
```

$T_1 = a - b \rightarrow s$

$T_2 = c \cdot d$

$T_3 = T_2 + x \rightarrow T$

$T_4 = T_3 - T_1 \rightarrow T$

$T_1 \rightarrow s$

$T_2 \rightarrow T$

$T_3 \rightarrow T$

$T_4 \rightarrow T$

$T_4 \rightarrow s$