11.05

1(a) Een reguliere definitie is een reeks definities van de volgende vorm

$d_1 \to r_1$
$d_2 \to r_2$
⋮
$d_n \to r_n$

(hierin is elke d_i een nieuw symbool.)

voor zekere $n \geq 1$. Als het oorspronkelijke alfabet Σ is, dan is, voor $i = 1, \ldots, n$, r_i een reguliere expressie over $\Sigma \cup \{d_1, \ldots, d_{i-1}\}$. Ofwel in de definitie van d_i mogen we de voorgaande d_j's gebruiken.

11.10

(b)

letter → a | b | .. | z | A | B | .. | Z
digit → 0 | 1 | .. | 9
identifier → (letter | _) (letter | digit | _)*

Ofwel, we beginnen met een letter of underscore, en daar mogen nul of meer letters, cijfers, en/of underscores achter staan

11.14

(c)

```
for i = 1 to n do
{ vervang in r_i elk voorkomen van d_j met j < i
    door r_j'                    // maakt niet uit in welke volgorde
  noem het resultaat r_i'        // de d_j's worden vervangen.
}
```

Dan is r_n' een reguliere expressie over Σ die equivalent is met d_n.

11.19

2(a) Voor elk tweetal verschillende producties $A \to \alpha$ en $A \to \beta$ (voor dezelfde nonterminal dus) moet gelden:

* $\text{FIRST}(\alpha) \cap \text{FIRST}(\beta) = \emptyset$
  (in het bijzonder mogen ze niet allebei ϵ bevatten)
* als $\epsilon \in \text{FIRST}(\alpha)$, dan is ook $\text{FOLLOW}(A) \cap \text{FIRST}(\beta) = \emptyset$

11.22/11.23

(b) G bevat zowel directe linksrecursie (bij nonterminal A) als indirecte linksrecursie ($S \Rightarrow A \ldots \Rightarrow B \ldots \Rightarrow S \ldots$)

* We ordenen de nonterminals in de volgorde S, A, B
* De producties van S 'wijzen alleen naar voren', dus die zijn OK.
* De producties van A 'wijzen naar voren' ($A \to Bb$) of naar zichzelf ($A \to Ac$). We elimineren de directe linksrecursie door de twee producties voor A te vervangen door:

$A \to BbA'$ $\quad$ $A' \to cA' \mid \epsilon$

* De productie van B 'wijst terug', en we vervangen haar
  eerst door $B \to AcBa \mid bAa$
  vervolgens (omdat $B \to AcBa$ ook nog 'terugwijst') door
  $B \to BbA'cBa \mid bAa$
  en ten slotte (omdat er nu nog directe linksrecursie is) door
  $B \to bAaB' \qquad B' \to bA'cBaB' \mid \epsilon$

11.33

G' bevat dus de volgende producties:

$S \to AcB \mid bA$
$A \to BbA'$
$A' \to cA' \mid \epsilon$
$B \to bAaB'$
$B' \to bA'cBaB' \mid \epsilon$

In het resultaat is geen linksfactorisatie mogelijk.

11.35 / 11.40

(c)

X	FIRST(X)	FOLLOW(X)
S	b	$
A	b	c, $, a
A'	c, ε	c, $, a
B	b	$, b, a
B'	b, ε	$, b, a

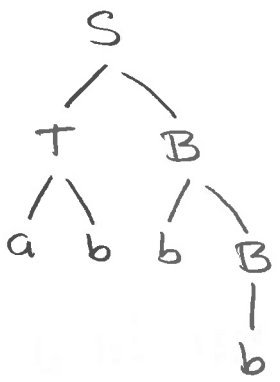
11.43

(d) Bij het bepalen van FIRST(S) merkten we al dat $FIRST(AcB) = FIRST(bA) = \{b\}$
Voor de twee producties $S \to AcB$ en $S \to bA$ geldt dus niet dat $FIRST(AcB) \cap FIRST(bA) = \emptyset$.
Volgens (a) is G' dus niet LL(1).

11.47

11.52 / 12.00

3(a)



12.01

(b)

States on stack	Corresponding symbols on stack	Input	Action
0	$	abbb $	shift (6)
06	$a	bbb $	shift 9
069	$ab	bb $	reduce naar T→ab (3)
02	$T	bb $	shift 4
024	$Tb	b $	shift 4
0244	$Tbb	$	reduce naar B→b (5)
0245	$TbB	$	reduce naar B→bB (4)
023	$TB	$	reduce naar S→TB (1)
01	$S	$	accept

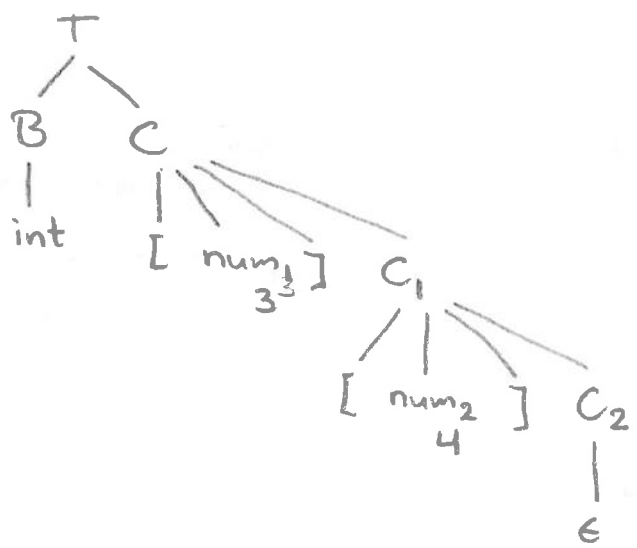
12.10

4 (a)

De width van een type is de hoeveelheid bytes die (een variabele van) dat type in beslag neemt.

12.12 / 12.14

(b)



12.18 / 12.26

(c) $C \rightarrow \epsilon$ { C.type = t; C.width = w; }

$C \rightarrow [num]\, C_1$ { C.type = array(num.val, C_1.type);
C.width = num.val * C_1.width;
}

12.31

(d) B.type = integer
B.width = 4
t = integer
w = 4
C_2.type = integer
C_2.width = 4
C_1.type = array(4, integer)
C_1.width = 4 * 4 = 16
C.type = array(3, array(4, integer))
C.width = 3 * 16 = 48
T.type = array(3, array(4, integer))
T.width = 48

12.38 / 12.39

5 (a) (i)

use(x, B) is het aantal gebruiken van variabele x binnen block B, vóórdat er een nieuwe waarde voor x (in block B) wordt berekend

12.43 / 12.50

live(x, B) is 1 als x live is aan het eind van block B en binnen B een nieuwe waarde heeft gekregen
0 anders.

12.54 (ii)

Wanneer x in B gebruikt wordt, voordat hij binnen B een nieuwe waarde krijgt, zou je x normaal gesproken in een register moeten laden.
Uitgaande van een allocatie algoritme dat variabele x in een register mag overschrijven als zijn huidige waarde ook nog elders (d.w.z. in geheugen) te vinden is, zul je x opnieuw in een register moeten laden als hij een tweede, derde, etc. keer wordt gebruikt in B vóórdat hij een nieuwe waarde krijgt. use(x, B) is dus een schatting voor het aantal loads dat

je in block B uitspaart, als je x vast in een register houdt gedurende loop L.
Normaal gesproken zou je x storen als hij live is aan het eind van block B en binnen B een nieuwe waarde heeft gekregen.
live(x,B) 'telt' dus het aantal stores dat je in block B uitspaart als je x vast in een register houdt. Omdat een store 'duurder' is dan een load, vermenigvuldigen we live(x,B) met 2.
We houden x gedurende de hele loop vast in een register. In elk block B in L kunnen we dus loads en stores uitsparen.
De bijdragen van de verschillende blocks tellen we bij elkaar op.

13.07
(b)
Een nadeel van het reserveren van een register voor een variabele is dat je minder registers over houdt voor de andere variabelen. Daardoor moet je voor de andere variabelen wellicht meer loads en stores uitvoeren.

13.09
6 (a)
De reaching definitions op een bepaald punt in een programma zijn de definities (toekenningen van waardes aan variabelen) die nog geldig kunnen zijn op dat punt in het programma.

13.11 / 13.12
(b)
$gen_d = \{d\}$
$kill_d = \{$alle andere definities van variabele u in het programma$\}$

13.14 / 13.16 / 13.23
(c)

```
OUT [ENTRY] = ∅
for...
  OUT [B] = ∅

while ...
  for ...
  {  IN [B] = ∪ predecessors P of B OUT[P]

     OUT [B] = gen_B ∪ ( IN [B] - kill_B )
  }
```

Hierin is gen_B de verzameling van definities in B die niet later in B gekilled worden door een andere definitie van dezelfde variabele, en is $kill_B = \cup_{\text{definities } d \text{ in } B} kill_d$

13.30

Uitwerking tentamen Compilerconstructie, donderdag 19 december 2019

13.32

(d)

B	gen_B	$kill_B$
B_1	$\{d_2, d_3\}$	$\{d_1, d_3, d_4, d_6, d_8, d_9\}$
B_2	$\{d_4\}$	$\{d_1, d_3, d_8\}$
B_4	$\{d_5\}$	$\{d_7\}$
B_3	$\{d_6, d_7\}$	$\{d_2, d_5, d_9\}$
B_5	$\{d_8, d_9\}$	$\{d_1, d_2, d_3, d_4, d_6\}$

13.37 / 13.40

(e)

We lopen de blokken af in de volgorde B_1, B_2, B_4, B_3, B_5, EXIT

B	$OUT[B]^0$	$IN[B]^1$	$OUT[B]^1$
B_1	$\emptyset$	$\emptyset$	$\{d_2, d_3\}$
B_2	$\emptyset$	$\{d_2, d_3\}$	$\{d_2, d_4\}$
B_4	$\emptyset$	$\{d_2, d_3\}$	$\{d_2, d_3, d_5\}$
B_3	$\emptyset$	$\{d_2, d_3, d_4, d_5\}$	$\{d_3, d_4, d_6, d_7\}$
B_5	$\emptyset$	$\{d_2, d_3, d_5\}$	$\{d_5, d_8, d_9\}$
EXIT	$\emptyset$	$\{d_5, d_8, d_9\}$	$\{d_5, d_8, d_9\}$

B	$IN[B]^2$	$OUT[B]^2$
B_1	$\{d_3, d_4, d_5, d_6, d_7, d_8, d_9\}$	$\{d_2, d_3, d_5, d_7\}$
B_2	$\{d_2, d_3, d_5, d_7\}$	$\{d_2, d_4, d_5, d_7\}$
B_4	$\{d_2, d_3, d_5, d_7\}$	$\{d_2, d_3, d_5\}$
B_3	$\{d_2, d_3, d_4, d_5, d_7\}$	$\{d_3, d_4, d_6, d_7\}$
B_5	$\{d_2, d_3, d_5\}$	$\{d_5, d_8, d_9\}$
EXIT	$\{d_5, d_8, d_9\}$	$\{d_5, d_8, d_9\}$

In de derde iteratie verandert er niets meer.

13.53