

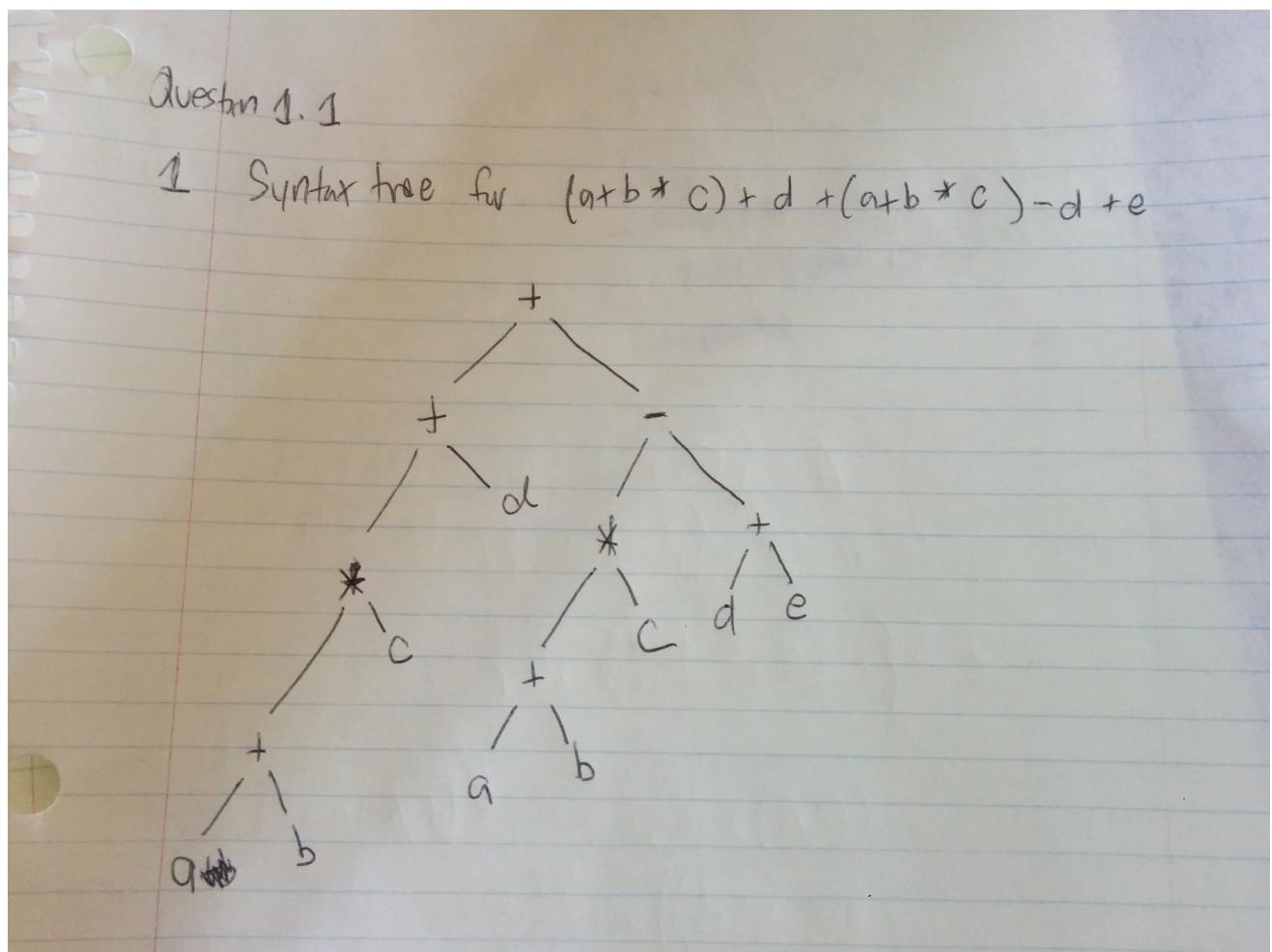
Homework 7

Abhi Agarwal

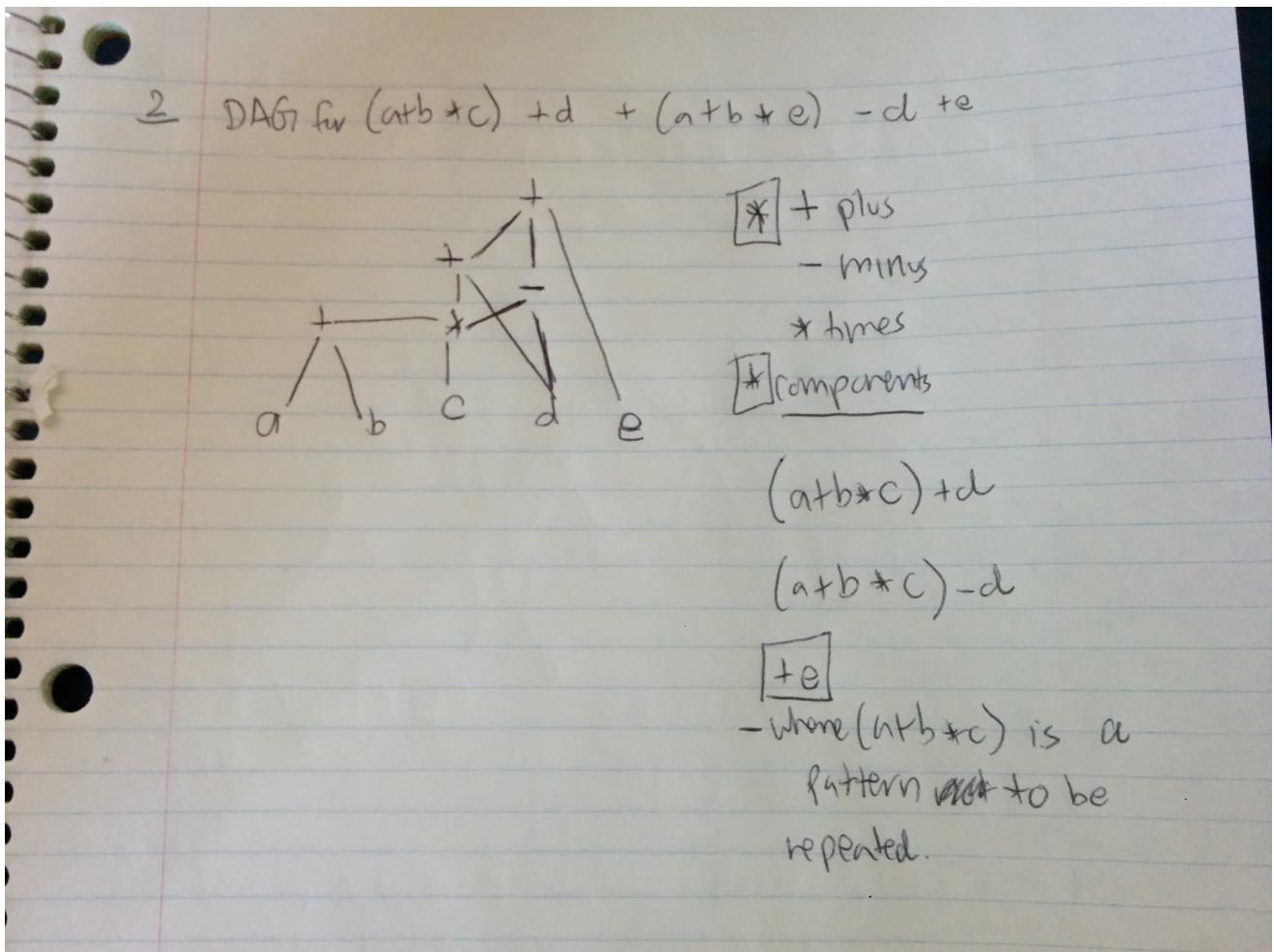
1 Representation

1.1 Question 1.1

1.1.1 Syntax Tree



1.1.2 DAG (Directed Acyclic Graph) representation



1.1.3 Three-Address Code representation

- where $(wkb+e)$ is a pattern ~~not~~ to be repeated.

3 Three - Address Code representation

$$\begin{aligned}t_1 &= a+b \\t_2 &= t_1 * c \\t_3 &= t_2 + d \\t_4 &= t_2 - d \\t_5 &= t_3 + t_4 + e\end{aligned}$$

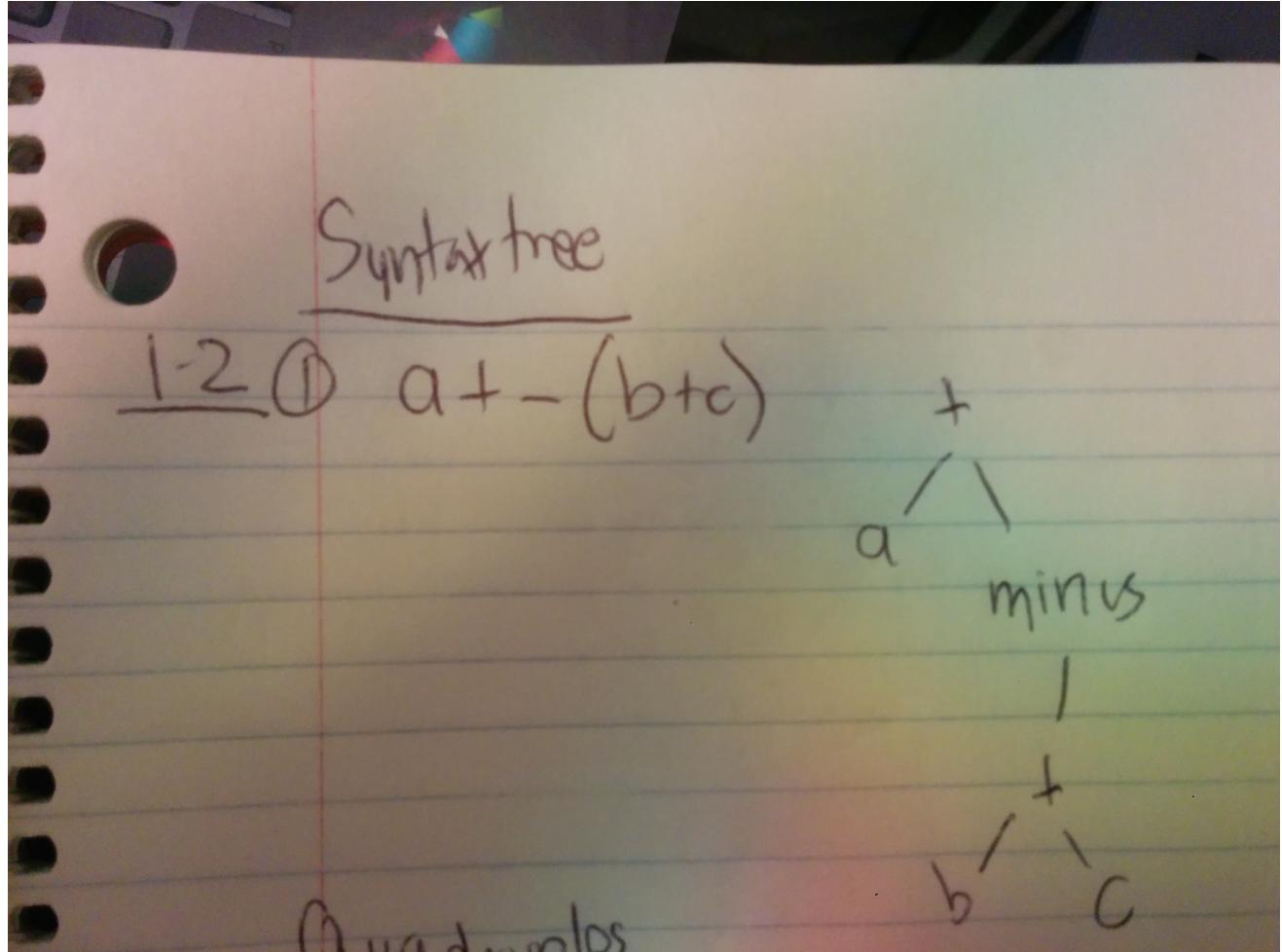
Where t_5 is final answer

```
graph TD; t1[t1] --- a[a]; t1 --- b[b]; t2[t2] --- c[c]; t2 --- d[d]; t3[t3] --- t2; t4[t4] --- t2; t5[t5] --- t3;
```

1.2 Question 1.2

1.2.1 Syntax Tree

Please change the word "minus" to an actual minus (-) (My mistake I looked at this later)



1.2.2 Quadruples

Quadruples

Op	Arg1	Arg2	Result
0	+ b	c	t_1
1	minus t_1		t_2
2	+ a	t_2	t_3

$t_1 = b + c$
 $t_2 = -t_1$
 $t_3 = a + t_2$

1.2.3 Triples

③ Triples

	OP	Arg1	Arg2	
0	+	b	c	
1	minus	(0)		
2	+	a	(1)	

Result referred to by position

```

    . + \
    a   \ minus
        | 
        + \
        b   \ c
  
```

1.2.4 Indirect Triples

Please change the 35, 36, 37 to 0, 1, 2 instead as that's arbitrary and I misunderstood it on the slides.

(4) <u>Indirect triples</u>				
	instuch	op	arg1	arg2
35	(0)	0	b	c
36	(1)	1	minus	(0)
37	(2)	2	a	(1)
.				
.				
.				

2 Translation

2.1 Question 2.1

2.1.1 Add a translation rule for the following expression production:
 $E \rightarrow E_1 * E_2$

① Production	Rules
$S \Rightarrow id = E_1 ; S_2$ ϵ	$S \cdot code = E_1 \cdot code [id' = 'E_1 \cdot addr] S_2 \cdot code$ $S_2 \cdot code = []$
$E \Rightarrow E_1 + E_2$	$E_1 \cdot e = E \cdot addr = newTemp()$ $E \cdot code = E_1 \cdot code E_2 \cdot code [E \cdot addr = E_1 \cdot addr + E_2 \cdot addr]$
$E \Rightarrow E_1 * E_2$	$E_1 \cdot e = E \cdot addr = newTemp()$ $E \cdot code = E_1 \cdot code E_2 \cdot code [E \cdot addr = E_1 \cdot addr * E_2 \cdot addr]$
$I - E_1$ ⋮	...

2.2 Question 2.2

2.2.1 Explain how the code is expected to evaluate, assuming parameters are passed call-by-value.

Taken from the slides: If parameters are passed call-by-value then the contract is:

$E_1.\text{code}$, ..., $E_n.\text{code}$ are evaluated before results placed into temporary

$E_1.\text{addr}$, ..., $E_n.\text{addr}$

hello

2.3 Question 2.3

2.3.1 $S \rightarrow \text{for}(S_1; B; S_2) S_3$

Extending the code from the book

Production	Semantic Rules
$S \rightarrow \text{for}(S_1; B; S_2) S_3$	$S_1.\text{next} = \text{newlabel()}$
	B.true = newlabel()
	$B.\text{false} = S.\text{next}$
	$S_2.\text{next} = S_1.\text{next}$
	$S_3.\text{next} = \text{newlabel()}$
	$S.\text{code} = S_1.\text{code}$
	$\quad \quad \quad \text{label}(S_1.\text{next}) \quad B.\text{code}$
	$\quad \quad \quad \text{label}(B.\text{true}) \quad S_3.\text{code}$
	$\quad \quad \quad \text{label}(S_3.\text{next}) \quad S_2.\text{code}$
	$\quad \quad \quad \text{gen('go to' } S_1.\text{next})$
referencing page 402.	

In some cases, the instruction S contains a jump to a label L . A jump to a jump to L from within S produces three-address code. The syntax-directed definition in Fig. 6.36-6.37 produces three-address code for boolean expressions in the context of if-, if-else-, and while-statements.

PRODUCTION	SEMANTIC RULES
$P \rightarrow S$	$S.next = newlabel()$ $P.code = S.code \parallel label(S.next)$
$S \rightarrow \text{assign}$	$S.code = \text{assign}.code$
$S \rightarrow \text{if } (B) S_1$	$B.true = newlabel()$ $B.false = S_1.next = S.next$ $S.code = B.code \parallel label(B.true) \parallel S_1.code$
$S \rightarrow \text{if } (B) S_1 \text{ else } S_2$	$B.true = newlabel()$ $B.false = newlabel()$ $S_1.next = S_2.next = S.next$ $S.code = B.code$ $\parallel label(B.true) \parallel S_1.code$ $\parallel \text{gen('goto' } S.next)$ $\parallel label(B.false) \parallel S_2.code$
$S \rightarrow \text{while } (B) S_1$	$begin = newlabel()$ $B.true = newlabel()$ $B.false = S.next$ $S_1.next = begin$ $S.code = label(begin) \parallel B.code$ $\parallel label(B.true) \parallel S_1.code$ $\parallel \text{gen('goto' } begin)$
$S \rightarrow S_1 S_2$	$S_1.next = newlabel()$ $S_2.next = S.next$ $S.code = S_1.code \parallel label(S_1.next) \parallel S_2.code$

Figure 6.36: Syntax-directed definition for flow-of-control statements

We assume that $newlabel()$ creates a new label each time it is called, and $label(L)$ attaches label L to the current node.