The German University Cairo **June 20, 2011**
Faculty of Media Engineering and Technology
Prof. Dr. Carmen Gervet

Bar Code

CSIS1003 – Compiler

Final Exam

Instructions: Please Read Carefully Before Proceeding.

1. The allowed time for this exam is **3 hours** (180 minutes).
2. (non-programmable) calculators are allowed.
3. No books or other aids are permitted for this test.
4. This exam booklet contains 17 pages, including this one. Three extra sheets of scratch paper are attached and have to be kept attached. **Note that if one or more pages are missing, you will lose their points. Thus, you must check that your exam booklet is complete**.
5. Please write your solutions in the space provided. If you need more space, **please use the back of the sheet containing the problem or on the three extra sheets and make an arrow indicating that.**
6. When you are told that time is up, please stop working on the test.

All the best.

Please, do not write anything on this page.

Exercise	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**Total**
Maximum Marks	3	3	2	2	7	3	2.5	3	5	3	3	36.5
Earned Marks												

Exercise 1 First and Follow sets **(3 marks)**

Consider the following grammar with 2 missing productions:

$S \rightarrow aS \mid \dots$ (1)

$A \rightarrow \dots$(2) $\mid \varepsilon$

$X \rightarrow cS \mid \varepsilon$

$Y \rightarrow dS \mid \varepsilon$

$Z \rightarrow eS$

Fortunately we have the First and Follow sets for this grammar

	FIRST	FOLLOW
S	{a,b,c,d,e}	{\$} ∪Follow(X) ∪Follow(Y) ∪Follow(Z)
A	{c,d,e,ε}	{b}
X	{c, ε}	First(Y)/ ε ∪ First(Z)
Y	{d, ε}	First(Z)
Z	{e}	Follow(A)
a	{a}	First(S)
b	{b}	Follow(S)
c	{c}	First(S)
d	{d}	First(S)
e	{e}	First(S)

First and follow sets on terminals are built the same way, with the first being the terminal itself obviously!

Reconstruct the grammar by filling the missing productions (1) and (2). Justify your answer

(1) Ab because b is in follow of A and „b" is in first(S) and first (A) has c,d,e which are also in FIRST(S) 1.5 marks (1 if correct and no justification.)

(2) XYZ because first(A) contains c,d,e which can only be found in X,Y and Z . The order is justified by the follow set of X and Y and Z. 1.5 marks (1 if correct and no justification.)

Exercise 2 LL(1) **(3 marks)**

Give an LL(1) grammar for the following language:

All strings with i "a" followed by i+1 "bc" where i ≥0.

For example *bc, abcbc,* and *aabcbcbc* are all in this language.

S → aX bc | bc

1.5 mark

if erroneous grammar because it does not recognize the same language but "some idea" there, -0.5

if same language but more rules defined for the grammar full grade.

IF same language but grammar not LL(1) give 1 mark.

Prove that your grammar is LL(1).

The two bodies of the production headed with S do not share elements in their first set.

First(aXbc) ={a}, First (bc) = {b}

1.5 marks

Proof needs to be based on the grammar.

Exercise 3 Top-down parsing **(2 marks)**

Consider the following grammar over the alphabet {a,b,c}

$S \rightarrow Xa$

$X \rightarrow bX \mid Y$

$Y \rightarrow Zc$

$Z \rightarrow bZ \mid \varepsilon$

3.1 Prove that this grammar is not LL(1).

First(bX) = {b}

First(Y) = First(Z) + {c} = {b,c} non empty intersection

1 mark

3.2 It is possible to drop exactly one production (not two!!!) from this grammar to obtain a new grammar generating the same language, and that is an LL(1) grammar. Identify this production and prove that the resulting grammar is LL(1).

Z -> bZ does not add anything and now the first(Y)={c}

OR X -> bX

1 mark for rule and proof!

The table does not need to be built, just need the First sets of the productions with X as head.

If problem identified but wrong production remove. Count 0.25 out of 1.

Remove -0.5 if no proof but correct rule removed.

Exercise 4 bottom-up parsing **(2 marks)**

Prove that there is no grammar with a single production that has a reduce-reduce conflict under SLR(1) parsing rules.

By definition a reduce reduce conflict implies 2 productions minimum

2 marks if all correct.

1.5 marks if some idea regarding the impossibility of having a conflict with only one production but the justification is messy!

Exercise 5 Constructing LR tables (2+1+3+1=7 marks)

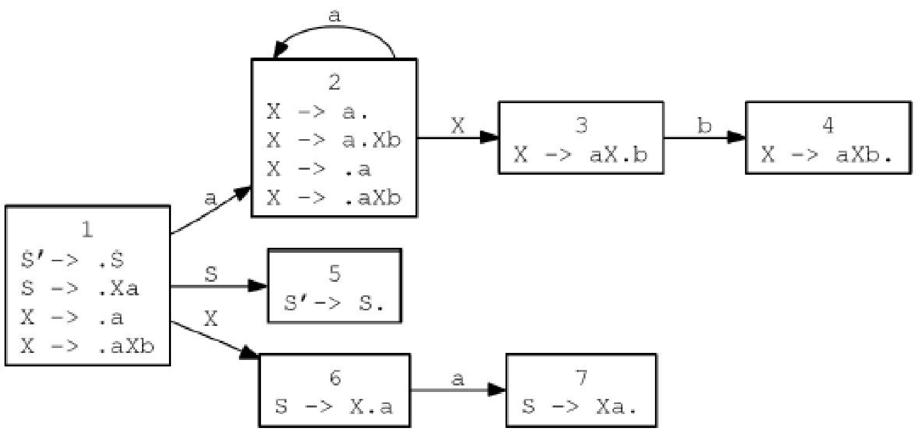
Consider the following CFG, which has the set of terminals T = {a,b} .

$S \rightarrow Xa$

$X \rightarrow a$

$X \rightarrow aXb$

5.1 Compute the set of LR(0) items for the augmented grammar and construct the DFA over these item sets.



2 MARKS : -0.10 per missing item in a set. -0.25 missing item set or wrong connection

5.2 Is the grammar LR(0)?

Grammar not LR(0) because shift-reduce conflict in 2.

1 mark (-0.5 if not justification but correct answer)

5.2 **Compute the follow sets of S and X and determine whether the grammar is SLR(1) or not. Justify your answer by building the SLR(1) parsing table.**

Follow(S) = {$}, Follow(X) = {a,b}. (0.5 marks)

Table (2 mark. -0.25 per erroneous entry!)

State	action			Goto	
	a	b	$	S	X
1	S2			5	6
2	S2/r2				3
3		S4			
4	R3	R3			
5			Acc		
6	S7				
7			R1		

Not SLR(1) shift reduce conflict in state 2.
0.5 mark for the answer.

5.3 **Suppose that the production $X \to \varepsilon$ is added to this grammar. Identify a reduce-reduce conflict in the resulting grammar under the SLR(1) rules.**

The item $X \to .$ will appear in the DFA state 2. Now in state 2 there will be two possible reductions on the lookahead symbol a: $X \to a$, $X \to .$
1 mark
remove 0.5 mark if erroneous statement but idea there.

Exercise 6 Bottom up parsing **(3 marks)**

Given the following LR(0) grammar:

E → (L)
E → a
L → L, E
L → E

Suppose that state 2 of the LR(0) DFA is the closure of the item E → (.L):

a) provide the full closure of this state

E → (.L)
L → .L, E
L → .E
E → .(L)
E → .a 1 mark

b) How many transitions are possible from this state? Justify your answer.

4 0.5 marks

c) Provide the configurations of the state reached from state 2 after recognizing the nonterminal L.

E → (L.)
L → L., E

0.5 mark

d) Without building the SLR(1) table, can you tell whether the grammar is SLR(1) or not? Justify your answer.

Yes it is because it is LR(0) so a fortiori it is SLR(1). They use the same DFA and there are no conflicts.

1 mark

Exercise 7 L-attributed and S-attributed grammars (2.5 marks)

Suppose we have a single production $A \rightarrow BCD$

Each of the four nonterminals A,B,C,D have two attributes: *s* is synthesized and *i* is inherited. For each of the set of rules below, tell whether (i) the rules are consistent with an S-attributed definition, (ii) the rules are consistent with an L-attributed definition, or (iii) whether the rules are consistent with neither. Justify your answers.

a) $A \rightarrow BCD$ {A.s = B.i + C.s}

S-attributed A.s computed from its children's attributes.

0.5 marks

b) $A \rightarrow BCD$ {A.s = B.i + C.s, D.i = A.i + B.s}

L-attributed: A.s computed from its children attribute, an D.i inherited from parent or left sibling attribute

0.5 marks -0.10 if attributes properties correct but answer not driven!

c) $A \rightarrow BCD$ {A.s = B.s + D.s}

S-attributed: A.s computed from its children's attributes

0.5 marks

d) $A \rightarrow BCD$ {A.s = D.i, B.i = A.s + C.s, C.i = B.s, D.i = B.i + C.i}

A.s is synthesized, B.i is neither since C.s is from a right sibling, C.i is inherited, and D.i is inherited.

So the grammar is neither

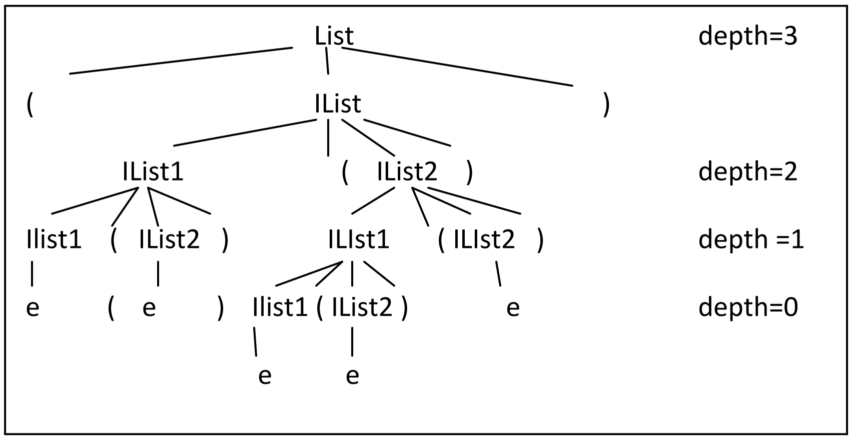
1 marks

Exercise 8 Semantic analysis **(3 marks)**

Consider the following attribute grammar that determines the depth of recursive lists.

List → (IList)	**{List.depth = IList.depth + 1;**
	Print "Depth:", List.depth }
IList → IList1 (IList2)	**{ if (IList1.depth > IList2.depth) :**
	IList.depth = IList1.depth
	else:
	IList.depth = IList2.depth + 1 }
IList → ε	**{IList.depth = 0 }**

8.1 Draw the parse tree and calculate the attributes for the input string: (( ) (( )( )))



1.5 marks for the correct tree, 1.5 for attributes

-0.25 per erroneous element

Exercise 9 Semantic analysis - Dependency graph **(2+1+1+1 = 5 marks)**

Consider the following attribute grammar with semantic rules:

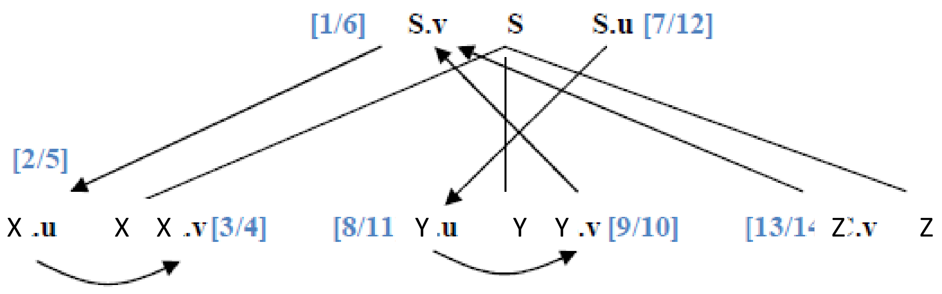
S → X Y Z **{ Y.u = S.u, S.v = Y.v +Z.v, X.u = S.v}**

X →a {X**.v = 2 * X.u}**

Y → b { **Y.v = Y.u }**

Z → c {Z**.v = 1}**

9.1 Draw the parse tree for the input 'a b c' (the only string for the language), and show the dependency graph for the associated attributes. Note that inherited attributes are conventionally written to the left of the node while synthesized attributes are posted to the right.



2 marks

9.2 Provide a topological sort to evaluate the attributes.

Z.v S.u Y.u Y.v S.v X.u X.v **1 mark**

9.3 Assume that S.u is assigned the value 5 before starting attribute evaluation, what will be the value of X.v when evaluation has terminated? (1 mark)

Answer: going through the ordering of the sort we have: Z.v S.u Y.u Y.v S.v X.u X.v
The computation of the attributes are: S.u = 5, Z.v = 1, Y.u = 5, Y.v = 5, S.v = 6, X.u = 6, X.v = 12

1 mark

9.4 Suppose now that the semantic rules were instead: (1 mark)

S → X Y Z **{ Y.u = S.u, S.v = Y.v +Z.v, X.u = S.v, Z.u = S.v}**

X →a {**X.v = 2 * X.u}**

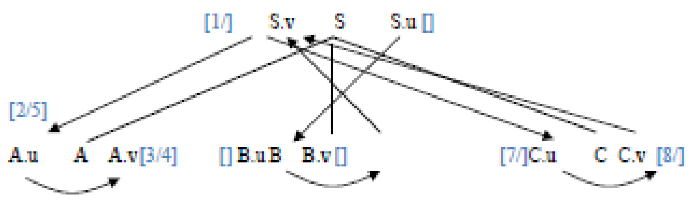
Y → b { **Y.v = Y.u }**

Z → c {**Z.v = Z.u-2}**

What value does X.v have after evaluation of all attributes, if the initial value of S.u is 3?

Explain why this result occurred? 1 mark

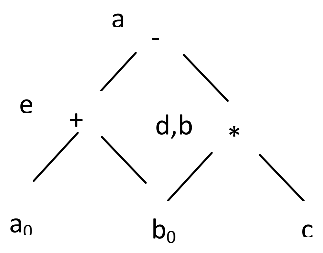
Answer:



There is a loop on S.v thus we cannot compute a topological sort and we cannot compute S.v needed for A.v

Exercise 10 DAG and Three-address code **(3 marks)**

The three-address code is a linearized version of the DAG. Consider the following DAG



10.1 Generate a corresponding three-address code.

e = a+b
d= b*c
b=b*c **/* we must have 'b' calculation value after that of d because 'd' uses 'b' value!! -0.5 if not**
a= e-b or a =e-d
1 mark

10.2 Only 'a' is live on exit from the block, simplify the three address code.

e = a+b
d= b*c or b=b*c
a= e-d a = e-b
1 mark

-0.25 per missing or erroneous statement

10.3 a,b,c are live on exit from the block, simplify the 3-address code.

e = a+b
b= b*c
a= e-b
1 mark

Exercise 11 Flow graph (3 marks)

Consider the following program.

```
1:   e:= 0
2:   b:= 1
3:   d:= 2
4:   a :=b+2
5:   c:= d + 5
6:   e := e+c
7:   f := a*a
8:   if f < c goto 11
9:   e := e+f
10: goto 12
11: e := e + 2
12: d := d + 4
13: b := b – 4
14: if b != d goto 4
15:
```

Compute the basic blocks for this 3-address code and draw the flow graph

$$\begin{tikzcd}
1,2,3 \arrow[rr] & & 4,5,6,7,8 \arrow[dl] \arrow[dr] & \\
& 9,10 \arrow[d] & & 11 \arrow[dll] \\
& 12,13,14 \arrow[uur, bend left] & & \\
& \text{Exit} & &
\end{tikzcd}$$

3 marks
-0.25 per erroneous block, arrow