

MIDTERM EXAM

- This test contains 7 questions with a total of 75 points
 - 1-2 5 points each
 - 3-4 10 points each
 - 5 15 points
 - 6 20 points
 - 7 10 points
- You have 50 minutes to complete this midterm.
- You may use a one sided 8.5" x 11" sheet of notes. You must turn in your notes with the test.
- You may **not** use your text, or any other reference material.
- You may remove the staple so you can see both parts of the last questions at the same time.
- Do not turn this page or turn over the exam until instructed to do so.

Hints

- Don't spend too much time on short answer questions. If you don't know the answer right away, move on and come back.
- There are some questions that will require more time to think and organize your thoughts. If you find yourself stuck on one of these questions, save it until you have answered all the questions you can answer quickly. If you spend too much time on one question you may get frustrated and that will impair your ability to answer the other questions.
- Don't turn your test in early. If you have time left over, use it to review your answers. Students who turn tests in early often make trivial mistakes that they would catch if they reviewed their answers.
- Students that get the highest grades usually go over their answers several times.
- The space below a question **does not** indicate how long of an answer I am expected. I've given you lots of room so you can show your work.
- Double check that you understand the question before you answer it. If you don't understand the question, ask me.

- 1) (5 points) What about the gpl grammar makes it ambiguous? Outline how you resolved the ambiguity in p3 (no details, just a brief overview of what you had to do).

The expressions and the if statement make gpl ambiguous.

The ambiguity was resolved by assignment precedence to the expression operators and by adding precedence to the if and if-else statements.

- 2) (5 points) What language does the following context-free grammar describe. The >, and - symbols are terminal tokens. The grammar generates strings that look like arrow symbols. You need to precisely describe the strings (the sorts of arrows).

arrow \rightarrow >> arrow >
arrow \rightarrow >> shaft >
shaft \rightarrow -
shaft \rightarrow - shaft -

Arrows in which there are twice as many tail feathers (> on the left) as barbs on the head of the arrow (> on the right) with a shaft of odd length.

- 3) (10 points) Demonstrate that the following grammar is ambiguous (ID, [, and] are terminal tokens). Explain your answer.

expr \rightarrow ID | expr [expr] | ID list
list \rightarrow [expr] list | ϵ

The following two left-most derivations derive the same string. This is sufficient to demonstrate that the language is ambiguous.

expr \Rightarrow expr [expr] \Rightarrow ID list [expr] \Rightarrow ID [expr] \Rightarrow ID [ID]

expr \Rightarrow ID list \Rightarrow ID [expr] list \Rightarrow ID [ID] list \Rightarrow ID [ID]

4) (10 points) Consider the language that contains all strings $a^i b^j$ where i is a positive odd number and j is a positive even number. For example: abb $aaabb$ $abbbb$ are all strings in the language. Can this language be defined by a context free grammar? If so, give the grammar. If not, can bison be used to parse the grammar? If so outline how.

Oops. I had intended this to be a harder problem. It started as a harder problem, but then I simplify it a little bit. Turns out I simplified it a lot.

The following context free grammar describes this language.

$$S \rightarrow aAbbB$$
$$A \rightarrow aaA \mid \epsilon$$
$$B \rightarrow bbB \mid \epsilon$$

5) (15 points) Eliminate the left recursion in the following grammar.

$S \rightarrow AaB \mid C$
 $A \rightarrow a \mid B$
 $B \rightarrow Bb \mid Sc$
 $C \rightarrow cS \mid \epsilon$

Step 1: Remove the non-immediate left recursion

Problem: $S \Rightarrow AaB \Rightarrow BaB \Rightarrow ScaB$ (remove non-immediate left recursion)

$S \rightarrow AaB \mid C$
 $A \rightarrow a \mid B$
 $B \rightarrow Bb \mid AaBc \mid Cc$ The “Sc” was replaced with “AaBc | Cc”
 $C \rightarrow cS \mid \epsilon$

Problem: $A \Rightarrow B \Rightarrow AaBc$ (remove non-immediate left recursion)

$S \rightarrow AaB \mid C$
 $A \rightarrow a \mid B$
 $B \rightarrow Bb \mid aaBc \mid BaBc \mid Cc$ The “AaBc” was replaced with “aaBc | BaBc”
 $C \rightarrow cS \mid \epsilon$

Step 2: Remove the immediate left recursion in the third production

$B \rightarrow Bb \mid aaBc \mid BaBc \mid Cc$ $\alpha_1 = b$ $\alpha_2 = aBc$ $\beta_1 = aaBc$ $\beta_2 = Cc$

$B \rightarrow aaBcB' \mid CcB'$
 $B' \rightarrow bB' \mid aBcB' \mid \epsilon$

Complete answer:

$S \rightarrow AaB \mid C$
 $A \rightarrow a \mid B$
 $B \rightarrow aaBcB' \mid CcB'$
 $B' \rightarrow bB' \mid aBcB' \mid \epsilon$
 $C \rightarrow cS \mid \epsilon$

6) (20 points) Compute the specified First and Follow sets for the following grammar, then construct a LL(1) parse table for the grammar. Put the sets in the First and Follow columns of the table. When filling in the table use the production number instead of writing out the production (for example, instead of writing “ $S \rightarrow \text{FOR} (A ; C ; A) S$ ” in the table write 2). Hint: calculate $\text{First}(A;)$, it is easy to miss this one. The symbols $() \{ \} ; = \text{FOR ID} .$ are all terminal tokens.

- (1) $S \rightarrow A ;$
- (2) $S \rightarrow \text{FOR} (A ; C ; A) S$
- (3) $S \rightarrow B$
- (4) $A \rightarrow V = E$
- (5) $A \rightarrow \epsilon$
- (6) $C \rightarrow E$
- (7) $C \rightarrow \epsilon$
- (8) $E \rightarrow V$
- (9) $V \rightarrow \text{ID } X$
- (10) $X \rightarrow . V$
- (11) $X \rightarrow \epsilon$
- (12) $B \rightarrow \{ L \}$
- (13) $L \rightarrow S L$
- (14) $L \rightarrow \epsilon$

	First	Follow		;	FOR	()	=	ID	.	{	}	\$
S	ID ; FOR {	\$ ID ; FOR { }	S	1	2				1		3		
A	ID ϵ	;))	A	5			5		4				
C	ID ϵ	;	C	7					6				
E	ID	;))	E						8				
V	ID	= ;)	V						9				
X	. ϵ	= ;)	X	11			11	11		10			
B	{	\$ ID ; FOR { }	B								12		
L	ID ; FOR { ϵ	}	L	13	13				13		13	14	

7) (10 points) Use the above parse table to parse the given input string. This string may or may not be a legal string in the language. There are enough lines in this table to either parse the string to determine it is an error. Put either the production or the production number in the action column. It will probably help prevent errors if you put the entire production.

Stack	Input	Action
\$S	FOR (ID = ID ; ID) { } \$	
\$S)A;C;A(FOR	FOR (ID = ID ; ID) { } \$	(2) $S \rightarrow \text{FOR} (A ; C ; A) S$
\$S)A;C;A((ID = ID ; ID) { } \$	Match FOR
\$S)A;C;A	ID = ID ; ID) { } \$	Match (
\$S)A;C;E=V	ID = ID ; ID) { } \$	(4) $A \rightarrow V = E$
\$S)A;C;E=X ID	ID = ID ; ID) { } \$	(9) $V \rightarrow \text{ID } X$
\$S)A;C;E=X	= ID ; ID) { } \$	Match ID
\$S)A;C;E=	= ID ; ID) { } \$	(11) $X \rightarrow \epsilon$
\$S)A;C;E	ID ; ID) { } \$	Match =
\$S)A;C;V	ID ; ID) { } \$	(8) $E \rightarrow V$
\$S)A;C;X ID	ID ; ID) { } \$	(9) $V \rightarrow \text{ID } X$
\$S)A;C;X	; ID) { } \$	Match ID
\$S)A;C;	; ID) { } \$	(11) $X \rightarrow \epsilon$
\$S)A;C	ID) { } \$	Match ;
\$S)A;E	ID) { } \$	(6) $C \rightarrow E$
\$S)A;V	ID) { } \$	(8) $E \rightarrow V$
\$S)A;X ID	ID) { } \$	(9) $V \rightarrow \text{ID } X$
\$S)A;X) { } \$	Match ID
\$S)A;) { } \$	(11) $X \rightarrow \epsilon$
		Error