

FINAL EXAM

- This test contains 9 questions with a total of 85 points
- 1-4 5 points each
- 5-8 10 points each
- 9 25 points
- You have 1 hour and 50 minutes to complete this exam.
- You may use **one** two-sided 8.5" x 11" sheet of notes (or any combination of paper that does not exceed 187 square inches of notes). You must turn in your notes with the test.
- You may **not** use the text, any electronic devices, or any other reference material.
- Do not turn this page or turn over the exam until instructed to do so.
- Turn off your cell phones.

Exam Reminders

- **If a question is unclear or you are not sure what I am asking, talk to me during the exam.**
- If I find a mistake during the exam, I will write it on the board (so if you see me writing on the board, look at what I am writing).
- Use additional sheets of paper if you need more room.
- **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 may 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 went back over their answers.
- Students that get the highest grades usually go over their answers several times.
- The space below a questions usually indicates how long of an answer I am expected. Sometimes I leave extra room so the next question will start on the next page.
- If your answer does not include anything you learned in this class, maybe you misunderstood the question.

1) (5 points) Describe where, when, and how array bounds are checked in gpl.

Arrays are referenced only in the Variable object. The Variable's get and set functions evaluate the index expression whenever a Variable is evaluated or set. After the expression is evaluated the actual variable name (e.g. "nums[42]") is constructed. If that name is not in the symbol table then the index is out of bounds.

2) (5 points) List the three types of analysis performed by a compiler. Give an example of each from your gpl interpreter.

Linear/Lexical: grouping digits into an integer

Hierarchical/Syntactic: does an assignment statement consist of: variable operator expression

Semantic: is the LHS of an assignment statement have a type that is compatible with the RHS

3) (5 points) In gpl all statements belong to a statement block. How many statement blocks are created for the following gpl code? List the statement(s) in each block. Number the blocks in terms of creation order. That is: indicate which block is the first block and the statements it contains. Then which block is the second block and the statements it contains and so on. Assume a,b,i are declared integers.

```
on space
{
  if (a < b)
    for (i = 0; i < 10; i += 1)
      {
        print("i = " + i);
      }
}
```

Block 1: event block for "on space"

contains the if statement

Block 2: the body block for the if statement in block 1

contains the for statement

Block 3: the initialization block for the for statement

contains the assignment statement $i = 0$

Block 4: the increment block for the for statement

contains the assignment statement $i += 1$

Block 5: the body block for the for statement

contains the print statement

4) (5 points) Draw the complete structure your program would create for the following code. Include all the objects involved starting with the Statement_block.

```
my_rect.x = nums[2 * k] + rects[i].x;
```

5) (10 points) Create a grammar that describes the language that contains the following strings:

$(a|b)^+ | (a|c)^+$ All strings that contain one or more a's and b's OR one or more a's and c's

$S \rightarrow B | C$

$B \rightarrow Ba | Bb | a | b$

$C \rightarrow Ca | Cb | a | c$

6) (10 points) Demonstrate that the following grammar is ambiguous. Explain your answer.

$S \rightarrow S 0 S 1 S$

$S \rightarrow S 1 S 0 S$

$S \rightarrow \epsilon$

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

$S \Rightarrow S0S1S \Rightarrow S0S1S0S1S \Rightarrow 0S1S0S1S \Rightarrow 01S0S1S \Rightarrow 010S1S \Rightarrow 0101S \Rightarrow 0101$

Note: after the first S is replaced with S 0 S 1, all the other S's are replaced with ϵ .

$S \Rightarrow S0S1S \Rightarrow 0S1S \Rightarrow 01S \Rightarrow 01S0S1S \Rightarrow 010S1S \Rightarrow 0101S \Rightarrow 0101$

7) (10 points) Consider the language that contains all non-empty strings composed of 0's or 1's where there are an equal number of 0's and 1's:

01	0110	101010
10	1100	...
1001	0011	
0101	000111	

It is not possible to create a non-ambiguous context free grammar for this language. However, it is possible to use bison to write a parser that can recognize if the input string is in the language or not in the language. Write bison pseudo-code for this parser (the pseudo-code must contain the grammar).

The trick is to use global variables and an action to check if there are an allowable number of 0's and 1's.

```
int num_0s = 0;
int num_1s = 0;

program:
  string
  {
    if (num_0s != 0 && num_0s == num_1s)
      print "input is in the language"
    else
      issue error
  }
;

string:
  string 0
  {
    num_0s++;
  }
|string 1
  {
    num_1s++;
  }
|
empty
;
```

8) ((10 points) Using the given grammar and parse table, fill in the Stack/Input/Action table with the moves an LR parser would take for the given input. The table contains enough rows. You may not need all the rows.

- (1) $E \rightarrow E + T$
- (2) $E \rightarrow T$
- (3) $T \rightarrow T * F$
- (4) $T \rightarrow F$
- (5) $F \rightarrow (E)$
- (6) $F \rightarrow id$

State	id	+	*	()	\$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Stack	Input	Action
0	id + (id * id) \$	s5
0 id 5	+ (id * id) \$	r6 F → id
0 F 3	+ (id * id) \$	r4 T → F
0 T 2	+ (id * id) \$	r2 E → T
0 E 1	+ (id * id) \$	s6
0 E 1 + 6	(id * id) \$	s4
0 E 1 + 6 (4	id * id) \$	s5
0 E 1 + 6 (4 id 5	* id) \$	r6 F → id
0 E 1 + 6 (4 F 3	* id) \$	r4 T → F
0 E 1 + 6 (4 T 2	* id) \$	s7
0 E 1 + 6 (4 T 2 * 7	id)\$	s5
0 E 1 + 6 (4 T 2 * 7 id 5)\$	r6 F → id
0 E 1 + 6 (4 T 2 * 7 F 10)\$	r3 T → T * F
0 E 1 + 6 (4 T 2)\$	r2 E → T
0 E 1 + 6 (4 E 8)\$	s11
0 E 1 + 6 (4 E 8) 11	\$	r5 F → (E)
0 E 1 + 6 F 3	\$	r4 T → F
0 E + T 9	\$	r1 E → E + T
0 E 1	\$	accept

9) (25 points (10 for SLR, 15 for LALR)) Show that the following grammar is not SLR. When building the collection of items, always consider the grammar symbols in this order: {S E F a b c}. You may not need all the states in the given tables. Only fill in the part of the SLR table necessary to demonstrate the grammar is not SLR (you must explain your answer). On the next page fill in the entire LALR table.

- (0) $S' \rightarrow S$
- (1) $S \rightarrow aEa$
- (2) $S \rightarrow aFb$
- (3) $S \rightarrow bEb$
- (4) $S \rightarrow bFa$
- (5) $E \rightarrow c$
- (6) $F \rightarrow c$

State	a	b	c	\$	S	E	F
0							
1							
2							
3							
4							
5							
6	r5/r6	r5/r6					
7							
8							
9							
10							
11							
12							
13							

First show the grammar is not SLR(1)
 Calculate the set of LR(0) items (below).
 Fill in the table.

Follow(S) = {\$}
 Follow(E) = {a,b}
 Follow(F) = {a,b}

For states [6,a] and [6,b] we reduce with rule 5 on Follow E and on rule 6 on Follow(F)

Thus we have reduce/reduce conflicts

Thus this grammar is NOT SLR

I0: $S' \rightarrow \cdot S$
 $S \rightarrow \cdot aEa$
 $S \rightarrow \cdot aFb$
 $S \rightarrow \cdot bEb$
 $S \rightarrow \cdot bFa$

I3: $S \rightarrow b \cdot Eb$
 $S \rightarrow b \cdot Fa$
 $E \rightarrow \cdot c$
 $F \rightarrow \cdot c$

I7: $S \rightarrow bE \cdot b$

I8: $S \rightarrow bF \cdot a$

I9: $S \rightarrow aEa \cdot$

I1: $S' \rightarrow S \cdot$

I4: $S \rightarrow aE \cdot a$

I10: $S \rightarrow aFb \cdot$

I2: $S \rightarrow a \cdot Ea$
 $S \rightarrow a \cdot Fb$
 $E \rightarrow \cdot c$
 $F \rightarrow \cdot c$

I5: $S \rightarrow aF \cdot b$

I11: $S \rightarrow bEb \cdot$

I6: $E \rightarrow c \cdot$
 $F \rightarrow c \cdot$

I12: $S \rightarrow bFa \cdot$

9) continued (15 points for this part) show all states

- (0) $S' \rightarrow S$
- (1) $S \rightarrow aEa$
- (2) $S \rightarrow aFb$
- (3) $S \rightarrow bEb$
- (4) $S \rightarrow bFa$
- (5) $E \rightarrow c$
- (6) $F \rightarrow c$

State	a	b	c	\$	S	E	F
0	s2	s3			1		
1				acc			
2			s6			4	5
3			s9			7	8
4	s10						
5		s11					
6	r5	r6					
7		s12					
8	s13						
9	r6	r5					
10				r1			
11				r2			
12				r3			
13				r4			

No conflicts in table.
Grammar is LALR

I0: $S' \rightarrow \cdot S, \$$
 $S \rightarrow \cdot aEa, \$$
 $S \rightarrow \cdot aFb, \$$
 $S \rightarrow \cdot bEb, \$$
 $S \rightarrow \cdot bFa, \$$

I1: $S' \rightarrow S \cdot, \$$

I2: $S \rightarrow a \cdot Ea, \$$
 $S \rightarrow a \cdot Fb, \$$
 $E \rightarrow \cdot c, a$
 $F \rightarrow \cdot c, b$

I3: $S \rightarrow b \cdot Eb, \$$
 $S \rightarrow b \cdot Fa, \$$
 $E \rightarrow \cdot c, b$
 $F \rightarrow \cdot c, a$

I4: $S \rightarrow aE \cdot a, \$$

I5: $S \rightarrow aF \cdot b, \$$

I6: $E \rightarrow c \cdot, a$
 $F \rightarrow c \cdot, b$

I7: $S \rightarrow bE \cdot b, \$$

I8: $S \rightarrow bF \cdot a, \$$

I9: $E \rightarrow c \cdot, b$
 $F \rightarrow c \cdot, a$

I10: $S \rightarrow aEa \cdot, \$$

I11: $S \rightarrow aFb \cdot, \$$

I12: $S \rightarrow bEb \cdot, \$$

I13: $S \rightarrow bFa \cdot, \$$