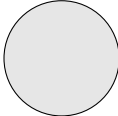
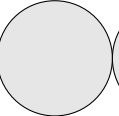
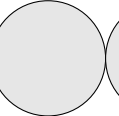
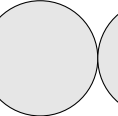
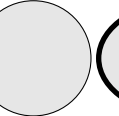
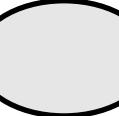


page 1	page 2	page 3	page 4	page 5	Total / 52	<i>Please print clearly :</i>
						<b>Name :</b>
						<b>Login :</b> @ucsc.edu

*No books ; No calculator ; No computer ; No email ; No internet ; No notes ; No phone. Neatness counts ! Do your scratch work elsewhere and enter only your final answer into the spaces provided.*

1. Given the grammar presented here, and using the style from the LALR(1) handout :
  - (a) Construct the characteristic finite state machine (CFSM), sets of items and transition diagram, showing shifts, reductions, and acceptance. **[6✓]**
  - (b) Construct the FOLLOW sets. **[3✓]**
  - (c) Answer *yes* or *no* to each of the following questions : **[1✓]**

Is the grammar LR(0) ? \_\_\_\_\_ Is the grammar SLR(1) ? \_\_\_\_\_

- |    |                         |
|----|-------------------------|
| 0. | $S \rightarrow \$ M \$$ |
| 1. | $M \rightarrow M E$     |
| 2. | $M \rightarrow$         |
| 3. | $E \rightarrow ( M )$   |
| 4. | $E \rightarrow * E$     |
| 5. | $E \rightarrow x$       |

2. Translate the C code on the left into icode as per the project 5 specs. Do not write any declarations for temporaries, just use them and assume they are of the correct type. As far as is possible, write the icode in the right box parallel to the C code. Note : we are using some constructs not available in c0. [7✓]

	<code>int compare (int, int);</code>	(Do not translate this line.)
1.	<code>int shift (</code>	
2.	<code>int *array,</code>	
3.	<code>int slot,</code>	
4.	<code>int saved</code>	
5.	<code>) {</code>	
6.	<code>while (slot &gt; 0) {</code>	
7.	<code>if (compare (</code>	
8.	<code>array [slot - 1],</code>	
9.	<code>saved) &lt;= 0)</code>	
10.	<code>break;</code>	
11.	<code>array[slot] =</code>	
12.	<code>array[slot - 1];</code>	
13.	<code>--slot;</code>	
14.	<code>}</code>	
15.	<code>return slot;</code>	
16.	<code>}</code>	

3. In the following table, assuming the function **shift** owns block 4, write the mangled names of all identifiers in the program under the heading *Identifier* and the associated attributes in the adjacent box labelled *Attributes*. Choose some combination of attributes from the following list: **int**, **char**, **void**, **array[n]**, **ptr(n)**, **function**, **lvalue**, **global**, **local**, **param**. For pointers, *n* is the number of levels, and for arrays, *n* is the dimension. [3✓]

Identifier	Attributes

Identifier	Attributes

4. For the program on the previous page :
- (a) Draw a graph of basic blocks. Draw a circle for each basic block in the program, and inside the circle write the numbers in the left column which refer to lines of code belonging to that basic block. Using the capital letters A, B, C, . . . , write one letter in each circle. **[2✓]**
  - (b) Draw a depth first spanning tree using the letters from the first diagram, not the numbers. **[1✓]**
  - (c) Draw a dominator tree and write an asterisk next to the head of the natural loop. Inside each circle, write the letters used in the previous question. **[2✓]**
  - (d) Using the letters, what blocks are part of the natural loop ? **[1✓]**
5. Using the general rules of project 3, draw an abstract syntax tree of that part of the AST for the previous question which is rooted at **while**. That is, ignore everything except for the **while** node and all of its descendants.. Make a reasonable choice for those elements not part of c0. **[4✓]**

Multiple choice. To the *left* of each question, write the letter that indicates your answer. Write **Z** if you don't want to risk a wrong answer. Wrong answers are worth negative points. **[11✓]**

number of correct answers		$\times 1 =$	$= a$
number of wrong answers		$\times \frac{1}{2} =$	$= b$
number of missing answers		$\times 0 =$	$0$
column total $c = \max(a - b, 0)$	11		$= c$

- Which of the following items will cause a reduction action to be added to the state ?
  - $E \rightarrow \bullet E + T$
  - $E \rightarrow E \bullet + T$
  - $E \rightarrow E + \bullet T$
  - $E \rightarrow E + T \bullet$
- The **flex** regex “dot star” ( $\cdot^*$ ) is equivalent to :
  - $[/n^*]^*$
  - $[\backslash n^*]^*$
  - $[\wedge n]^*$
  - $[\wedge \backslash n]^*$
- Which statement is true ?
  - All LR(k) languages are unambiguous.
  - All context free languages are LR(k).
  - Some LR(k) languages are ambiguous.
  - Some regular languages are not LR(k).
- The grammar :
 
$$A \rightarrow x$$

$$A \rightarrow y$$
  - is both LR(0) and SLR(1).
  - is LR(0) but not SLR(1).
  - is not LR(0) but is SLR(1).
  - is neither LR(0) nor SLR(1).
- Which **bison** grammar is ambiguous ?
  - $e : I X I \mid I ;$
  - $e : I X e \mid I ;$
  - $e : e X I \mid I ;$
  - $e : e X e \mid I ;$
- Which **bison** grammar is unambiguous, allows an arbitrary number of occurrences of **I**, and makes **X** right associative ?
  - $e : I X I \mid I ;$
  - $e : I X e \mid I ;$
  - $e : e X I \mid I ;$
  - $e : e X e \mid I ;$
- The string `xxxxxxxx` is a sentence in the language defined by which grammar ?
  - $A \rightarrow A x$   
 $A \rightarrow y$
  - $A \rightarrow A y$   
 $A \rightarrow x$
  - $A \rightarrow x A$   
 $A \rightarrow y$
  - $A \rightarrow y A$   
 $A \rightarrow x$
- Which answer has only one basic block ?
  - $a = b + f(x); g(h(y)); m = n;$
  - $a = b < c ? d > e : f == g;$
  - `for (i = 0; i < n; ++i) s += a[i];`
  - `if (a[i] > 3) t = false;`
- An AST node with what keyword will require data associated with code generation of a jump target using information not in the node being considered ?
  - `continue`
  - `if`
  - `return`
  - `while`
- How many tokens are there in the following line of Java code ?
 

```
out.printf ("Hello%n"); // Say hello.
```

  - 7
  - 8
  - 9
  - 10
- Which **bison** grammar will parse input that causes this program to return success ? Assume **yylex** returns all non-whitespace characters as themselves.
 

```
int main () { int n = 0; int c;
  while ((c = getchar()) != EOF) {
    if (isspace(c)) continue;
    switch (c) {
      case '(': ++n; break;
      case ')': if (--n < 0) return 1; break;
      default: return 1;
    }
  }
  return 0;
}
```

  - $b : '(' ')' b \mid ;$
  - $b : '(' b ')' b \mid ;$
  - $b : '(' b ')' \mid ;$
  - $b : '(' b \mid b ')' \mid ;$

Multiple choice. To the *left* of each question, write the letter that indicates your answer. Write **Z** if you don't want to risk a wrong answer. Wrong answers are worth negative points. **[11✓]**

number of correct answers		$\times 1 =$	$= a$
number of wrong answers		$\times \frac{1}{2} =$	$= b$
number of missing answers		$\times 0 =$	$0$
column total $c = \max(a - b, 0)$	11		$= c$

- Which kind of storage management provides for the fastest version of **malloc**?
  - copying collector with semispaces
  - mark and sweep collector
  - reference counting collector
  - using **free** explicitly
- Which grammar is unambiguous, recognizes an arbitrarily large number of occurrences of **x**, and uses the smallest amount of parsing stack space?
  - $e : x x \mid x ;$
  - $e : x e \mid x ;$
  - $e : e x \mid x ;$
  - $e : e e \mid x ;$
- In Java, genericity with primitive data types is accomplished by :
  - automatic boxing and unboxing.
  - determining all types at runtime.
  - recompilation for each generic instantiation.
  - tagging integers and pointers.
- What is the appropriate semantic action to attach to the **bison** rule  
`expr : '(' expr ')'`
  - $\{ \$\$ = \$1; \}$
  - $\{ \$\$ = \$2; \}$
  - $\{ \$1 = \$\$; \}$
  - $\{ \$2 = \$1; \}$
- Given an NFA constructed via Thompson's construction, which is then converted into a DFA using the subset construction, but not minimized, the NFA will likely be [x] than the DFA, and when used in scanning, the NFA will likely run more [y] than the DFA.
  - [x] = larger, [y] = quickly
  - [x] = larger, [y] = slowly
  - [x] = smaller, [y] = quickly
  - [x] = smaller, [y] = slowly
- An LALR(1) shift/reduce conflict occurs when there is :
  - a reduction action whose lookahead set contains a nonterminal symbol which also labels an outgoing transition.
  - a reduction action whose lookahead set contains a terminal symbol which also labels an outgoing transition.
  - more than one reduction action in the same state and their lookahead sets are not disjoint.
  - no reduction action in a state and the symbol produced by the scanner does not appear on any outgoing transition.
- If the file **foo.sh** has as its first line  
`#!/bin/sh -x`  
 and it is executed with the command  
`foo.sh -bar baz`  
 then, when the executable binary **/bin/sh** starts up, the name of the script will be found in :
  - `argv[0]`
  - `argv[1]`
  - `argv[2]`
  - `argv[3]`
- If a string is used to subscript an array in Java, how will that error be detected?
  - by the scanner
  - by the parser
  - by the semantic analyzer
  - when the program is run
- In a local stack frame, the static link points at :
  - the call instruction which activated this function
  - the local frame of the caller
  - the local frame of the function in which the current function is nested
  - the object referred to as **this** in Java and **C++**
- Which **flex** regex will match one or more **a**'s followed by one or more **b**'s?
  - $(a|b)^+$
  - $a^*b^*$
  - $a^+b^+$
  - $a^+|b^+$
- What parsing action pushes the scanner's lookahead symbol onto the parsing stack and calls the scanner to replenish that symbol?
  - accept
  - error
  - reduce
  - shift