Computer Science 414
Spring 2015

Midterm 2 Practice Problems

We will go over solutions on Friday, April 25th

1. Give the LR(0) states and transitions, as well as the LR(0) parse table, for the following grammars

   (a)

Terminals	=	{id, ., [,], int }
Non-Terminals	=	$\{S, V\}$
Rules	=	(0) $S \to V\$$
	=	(1) $V \to$ id
		(2) $V \to V$. id
		(3) $V \to V$ [V]
		(4) $V \to V$ [int]
Start Symbol	=	S

   (b)

Terminals	=	{a, (,), \$}
Non-Terminals	=	$\{S', S, A\}$
Rules	=	(0) $S' \to S\$$
		(1) $S \to SA$
		(2) $S \to A$
		(3) $A \to (S)$
		(4) $A \to$a
Start Symbol	=	S'

2. Create a set of LR(0) items and transitions for the following grammar. Show that the grammar is not LR(0). Give the SLR(1) parse table for the grammar.

Terminals	=	{num, id, [,], \$}
Non-Terminals	=	$\{E', E, V\}$
Rules	=	(0) $E' \to E\$$
		(1) $E \to V$
		(2) $E \to$ num
		(3) $V \to$ id
		(4) $V \to V[E]$
Start Symbol	=	E'

3. Standard Java allows loop exits using the "break" and "continue" statements. A "break" statement leaves the current loop entirely, and a "continue" statement jumps to the end of the current loop. A break statement can only appear inside a loop or a switch statement, and a continue can only appear inside a loop. If we add "break" and "continue" to simpleJava, what changes must be made to the semantic analyzer to ensure that these statements only appear within loops?

4. When creating the Abstract Assembly for a function definition, Do we *need* to store the Stack Pointer on the stack? What about the Frame Pointer and Return Register? Explain.

5. Given the following class definitions, and the local variable declarations in the function foo:

```
class c1 {
   int w;
   boolean z;
}

class c2 {
   c1 y[];
   int z;
}

int foo(int x,y) {
   boolean a;
   boolean b[];
   c1 C;
   c2 D[];

  /*  Body of foo  */

}
```

Give the abstract assembly tree for each of the following statements, if they appeared in the body of foo:

(a) `y++;`

(b) `b[3] = false;`

(c) `C.w = x + y;`

(d) `D[x].z--;`

(e) `D[C.w].y[x].z = x > 3;`

(f) `while (x < 10) x = x + 3;`