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 =  
\begin{align*}  
(0) & \quad S \rightarrow V$ 
(1) & \quad V \rightarrow \text{id} 
(2) & \quad V \rightarrow V \cdot \text{id} 
(3) & \quad V \rightarrow V [ V ] 
(4) & \quad V \rightarrow V [ \text{int} ] 
\end{align*}  
Start Symbol = \text{S}  

(b)  
Terminals = \{a, (, ), $\}  
Non-Terminals = \{S', S, A\}  
Rules =  
\begin{align*}  
(0) & \quad S' \rightarrow SS$ 
(1) & \quad S \rightarrow SA 
(2) & \quad S \rightarrow A 
(3) & \quad A \rightarrow (S)  
(4) & \quad A \rightarrow a 
\end{align*}  
Start Symbol = \text{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.

Termins = \{num, id, [, ], $\}  
Non-Terminals = \{E', E, V\}  
Rules =  
\begin{align*}  
(0) & \quad E' \rightarrow ES$  
(1) & \quad E \rightarrow V  
(2) & \quad E \rightarrow \text{num}  
(3) & \quad V \rightarrow \text{id}  
(4) & \quad V \rightarrow V[E] 
\end{align*}  
Start Symbol = \text{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:

```java
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;