Use the following 3-address code and Java stack code instructions for answering code generation questions.

### 3-address Instruction

<table>
<thead>
<tr>
<th>3-addr Instruction</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>load R1 x</td>
<td>R1 ← x</td>
</tr>
<tr>
<td>store x R1</td>
<td>x ← R1</td>
</tr>
<tr>
<td>add R1 R2 R3</td>
<td>R1 ← R2 + R3</td>
</tr>
<tr>
<td>sub R1 R2 R3</td>
<td>R1 ← R2 - R3</td>
</tr>
<tr>
<td>mult R1 R2 R3</td>
<td>R1 ← R2 * R3</td>
</tr>
<tr>
<td>neg R1 R2</td>
<td>R1 ← -(R2)</td>
</tr>
</tbody>
</table>

### Java Stack Code

<table>
<thead>
<tr>
<th>Java Stack Code</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>nop</td>
<td>none</td>
</tr>
<tr>
<td>ldc_int c</td>
<td>push constant c onto stack</td>
</tr>
<tr>
<td>iload index(x)</td>
<td>push local variable X onto stack</td>
</tr>
<tr>
<td>istore index(x)</td>
<td>pop stack, store in local variable X</td>
</tr>
<tr>
<td>iadd</td>
<td>pop 2 elems off stack, add, push</td>
</tr>
<tr>
<td>isub</td>
<td>pop 2 elems off stack, subtract, push</td>
</tr>
<tr>
<td>imult</td>
<td>pop 2 elems off stack, multiply, push</td>
</tr>
<tr>
<td>ineg</td>
<td>pop stack, negate, push</td>
</tr>
<tr>
<td>goto L</td>
<td>jump to handle L</td>
</tr>
<tr>
<td>ifeq L</td>
<td>pop stack, jump to handle L if zero</td>
</tr>
<tr>
<td>if_icmpeq L</td>
<td>pop 2 elems, jump to L if equal</td>
</tr>
<tr>
<td>if_icmpgt L</td>
<td>pop 2 elems, jump to L if 1st greater</td>
</tr>
<tr>
<td>dup</td>
<td>duplicate top of stack</td>
</tr>
<tr>
<td>pop</td>
<td>pop top of stack</td>
</tr>
<tr>
<td>swap</td>
<td>swap top two positions of stack</td>
</tr>
</tbody>
</table>

1. **Run-time environment**

   (a) What is a frame used for?

   (b) Name six items of run-time information stored in a frame. For each item, identify whether its value is set before the procedure is called, during procedure execution, or right before procedure return.

   (c) Name one type additional type of information only stored in a frame for Java programs.

   (d) When can storage for a variable be allocated in a frame?

   (e) What advantage is obtained when allocating variables in a frame?

   (f) Name two advantages of managing memory allocation manually in the code.

   (g) Name two advantages of managing memory allocation automatically in the run-time system.

   (h) Name two methods of managing memory allocation automatically in the run-time system.

2. **Intermediate representations.**

   Consider the statements:

   - \( x := a + (b*a) \)
   - \( x := a - ((b + a) * c) \)

   (a) Translate each into an AST

   (b) Translate each into 3-address code

   (c) Translate each into Java stack code

   (d) Which representation is the most compact? Why?

   (e) Which representation is easy to manipulate? Why?

   (f) Which representation is hard to manipulate? Why?

   (g) Which representation is closest to the input program? Why?

3. **Code generation.**

   You are generating code for a Java stack machine. You are given the following grammar attributes and helper functions:

   **Attribute**
   - AstNode.code

   **Function**
   - genInst( X )
   - append( ... )

   **Holds**
   - list of instructions

   **Effect**
   - create new instruction X
   - returns handle to instruction

   **Concatenates lists of instructions**

   (a) What grammar actions needed to generate code for a C-style IF statement in the following production?

   \[
   \text{stmt} \rightarrow \text{IF ( exp ) stmtList } ; \\
   \{ \text{stmt.code} = \\text{??}; \}
   \]

   (b) What grammar actions needed to generate code for a C-style FOR loop in the following production?

   \[
   \text{stmt} \rightarrow \text{FOR (stmt ; exp ; stmt) stmt } ; \\
   \{ \text{stmt.code} = \\text{??}; \}
   \]

   (c) Write grammar actions needed to generate control code for an AND expression in the following production, using numerical value representation of booleans. Use _short circuiting_.

   \[
   \text{exp} \rightarrow \text{exp}_1 \text{AND exp}_2 \\
   \{ \text{exp.code} = \\text{??}; \}
   \]

   (d) Write grammar actions needed to generate control code for an NOR expression in the following production, using numerical value representation of booleans. Use _short circuiting_.

   \[
   \text{exp} \rightarrow \text{exp}_1 \text{NOR exp}_2 \\
   \{ \text{exp.code} = \\text{??}; \}
   \]

   (e) Write grammar actions needed to generate control code for a \( \geq \) (GEQ) expression in the following production, using numerical value representation of booleans.

   \[
   \text{exp} \rightarrow \text{exp}_1 \text{GEQ exp}_2 \\
   \{ \text{exp.code} = \\text{??}; \}
   \]
4. Complex code generation.

(a) Name two issues and solutions to generating code for function calls in C.

(b) Name two issues and solutions to generating code for array references in C.

(c) What code must the compiler generate for the code
   
   ```c
   int i, a[100];
   ...
   a[i + 5] = 4;
   ```

(d) What code must the compiler generate for the code
   
   ```c
   int foo(int x);
   ...
   x = foo(i + 2);
   ```
   
   i. Assuming foo() is call-by-value?
   ii. Assuming foo() is call-by-reference?