1. Write out the symbol table(s) (including any attributes that you think are important) for the following piece of code:

```c
int x;
float y;
string z = "hello";

void foo() {
    int x;
    float z;
}

void main() {
    int q;
    string p = "goodbye";
}
```

Answer:

Symbol table for GLOBAL:

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Value</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>x</td>
<td>—</td>
<td>0</td>
</tr>
<tr>
<td>float</td>
<td>y</td>
<td>—</td>
<td>4</td>
</tr>
<tr>
<td>string</td>
<td>z</td>
<td>&quot;hello&quot;</td>
<td>8</td>
</tr>
</tbody>
</table>

Symbol table for MAIN:

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Value</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>q</td>
<td>—</td>
<td>0</td>
</tr>
<tr>
<td>string</td>
<td>p</td>
<td>&quot;goodbye&quot;</td>
<td>4</td>
</tr>
</tbody>
</table>

Symbol table for FOO:

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Value</th>
<th>Offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>x</td>
<td>—</td>
<td>0</td>
</tr>
<tr>
<td>float</td>
<td>z</td>
<td>—</td>
<td>4</td>
</tr>
</tbody>
</table>

2. Explain the difference between an L-value and an R-value

Answer: An L-value is an address that can be stored to (i.e., a value that can
appear on the left-hand side of an assignment statement. An R-value is an actual data value (*i.e.*, a value that can appear on the right-hand side of an assignment statement).

3. ECE 468 student Cam Piler has come up with a code generation strategy for his project: while walking his Abstract Syntax Tree, whenever he sees a variable, he will immediately generate code to load its value into a temporary. He thinks this will make his compiler more efficient, because he generates code as soon as possible. Is he right? Why or why not?

**Answer:** This is less efficient. Recall that variables can appear on either the left or the right of an assignment statement. If a variable appears on the right-hand side, it is used as an R-value, and loading from it is the right thing to do. However, if the variable appears on the left-hand side, it is used as an L-value, and it should not be loaded from—we never actually need the value stored in the variable, we just want to store something to the address. Cam’s optimization will generate a lot of useless code.

4. Give the abstract syntax tree for the following expression (assume the usual order of operations):

\[(a + b) - (c * d)\]

**Answer:**

```
OP: –
   OP: +
   OP: *
      ID: a
      ID: b
      ID: c
      ID: d
```

The generated code for this expression is:

```
LOAD(A) T1
LOAD(B) T2
+ T1 T2 T3
LOAD(C) T4
LOAD(D) T5
* T4 T5 T6
– T3 T6 T7
```
5. Consider a three-address code specification with the following instructions:

- **LOAD(T1) T2** – load from the temporary or variable named T1 (which must be an L-value) and place the value into T2 (which will be an R-value).
- **STORE(T1) T2** – store the value in temporary T1 (which must be an R-value) into the variable T2 (which must be an L-value).
- **OP T1 T2 T3** – perform the operation \( T3 = T1 \ OP \ T2 \), where T1, T2 and T3 are R-values, and OP is either ADD, MUL or SUB.

Give the generated code for the expression in question 4. Also give the generated code for the following expressions. (If you create temporaries, call them T1, T2, T3, etc.)

(a) \( a + b + c + d \)

- LOAD(A) T1
- LOAD(B) T2
- + T1 T2 T3
- LOAD(C) T4
- + T3 T4 T5
- LOAD(D) T6
- + T5 T6 T7

(b) \( (a \ b) - c \)

- LOAD(A) T1
- LOAD(B) T2
- * T1 T2 T3
- LOAD(C) T4
- - T3 T4 T5

(c) \( a \ * \ (b - c) \)

- LOAD(A) T1
- LOAD(B) T2
- LOAD(C) T3
- - T2 T3 T4
- * T1 T4 T5