1. Call Stack

For the following sub-problems, consider the following program:

```c
void main() {
    int x;
    int y;
    float f;
    x = x + y * 2;
    y += foo(x, y);
    f = bar(x, y);
    ... // some computations
}

int foo(int a, int b) {
    float g;
    g = bar(a - b, a + b);
    return (int)g;
}

float bar(int r, int s) {
    float h;
    h = 3.14 * (r * s);
    return h;
}
```

(a) Assume your program is running on a machine with 4 registers, using callee saves. Assume address (i.e., pointers) are 8 bytes, floats are 4 bytes and ints are 4 bytes. Draw the complete stack (i.e., the stack including all active activation records) for the program right after foo has called bar, and before bar returns. For each slot in the stack, indicate what is stored there, and how much space that slot takes up.

(b) Now assume that your program uses caller saves instead. Draw the complete stack for the program right after main calls bar, and before bar returns.

Solution:
2. Type Checking

For this question, consider the same code as in Q1.

(a) Use the AST structure we developed for the course project, draw the AST for the line of code \( x = x + y \times 2 \); and show on the AST you drew how type information is assigned to each node.

**Solution:**

```
Assign [int = int]

Var: x [int]           BinOp: + [int]

Var: x [int]           BinOp: * [int]

Var: y [int]           Const: 2 [int]
```
(b) Repeat the process for the line \( f = \text{bar}(x, y); \).

**Solution:**

```
Assign [float = float]
  |  \\
Var: f [float]  Call: bar [float]
  |  \\
Var: x [int]  Var: y [int]
```

(c) Repeat the process for the line \( h = 3.14 \times (r \times s); \). Mark with a star where type checking would fail.

**Solution:**

Problem node marked in red.

```
Assign []
  \\
Var: h [float]  * BinOp * []
  \\
Const: 3.14 [float]  BinOp: * [int]
    \\
Var: r [int]  Var: s [int]
```