Problem Set 3

Fall 23

Due: Sunday, October 15th.

1. Call Stack

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

```
float bar(int r, int s) {
    float i;
    i = 20.0 + (r + s);
    return i;
}

int foo1(int a , int b) {
    float g;
    g = bar(b, a);
    return (int)g;
}

int foo2(int a , int b) {
    float h;
    h = bar(a, b);
    return (int)h;
}

void main() {
    int x;
    int y;
    float f;
    x = 1 + x * y;
    y = foo1(x, y) + foo2(x, y);
     ... //some computations
}
```

   (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 the first time* ***bar*** *has been called, 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 the second time* ***bar*** *has been called, and before* ***bar*** *returns.*

Hint: The left operand gets executed before the right operand for binary operations.

Solution:

a) Callee save

x (4 bytes)	
y (4 bytes)	
f (4 bytes)	
arg b for foo1 (4 bytes)	*main*
arg a for foo1 (4 bytes)	
return value for foo1 (4 bytes)	
main's old return address (8 bytes)	
main's old frame pointer (8 bytes)	
saved registers (4*4 bytes)	
g (4 bytes)	
arg s for bar (4 bytes)	*foo*1
arg r for bar (4 bytes)	
return value for bar (4 bytes)	
foo1's old return address (8 bytes)	
foo1's old frame pointer (8 bytes)	
saved registers (4*4 bytes)	*bar*
i (4 bytes)	

b) Caller Save

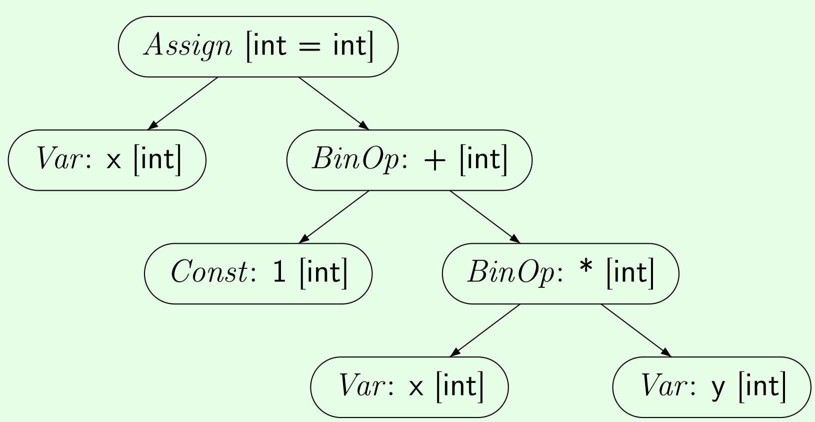
x (4 bytes)	
y (4 bytes)	
f (4 bytes)	
saved registers (4*4 bytes)	
arg b for foo2 (4 bytes)	*main*
arg a for foo2 (4 bytes)	
return value for foo2 (4 bytes)	
main's old return address (8 bytes)	
main's old frame pointer (8 bytes)	
h (4 bytes)	
saved registers (4*4 bytes)	
arg s for bar (4 bytes)	*foo*2
arg r for bar (4 bytes)	
return value for bar (4 bytes)	
foo2's old return address (8 bytes)	
foo2's old frame pointer (8 bytes)	
i (4 bytes)	*bar*

2. Type Checking

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

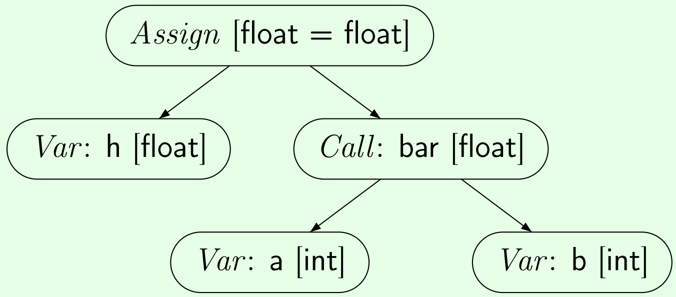
   (a) Use the AST structure we developed for the course project, draw the AST for the line of code x = 1 + x * y; and show on the AST you drew how type information is assigned to each node.

Solution:



(b) Repeat the process for the line h = bar(a, b);.

Solution:



(c) Repeat the process for the line i = 20.0 + (r + s);. Mark with a star where type checking would fail.

Solution:

Problem node marked in red.