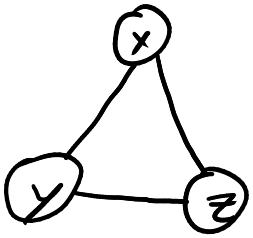
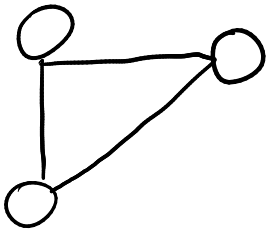
```
// code hoisting using
// busy expressions
while (...) {
        t2 = x + 1
        if (...) {
                t1 = y + 1
                //t2 = x + 1
                print(t1 + t2)
        } else {
                t1 = z + 1
                //t2 = x + 2
                write(t1 - t2)
        }
        x = input()
        y = input()
        z = input()
}
```

```
// reaching definitions
x = 3
x = x + y
print(x)
// more code
x = 4
x = x * y
print(x)

// transform to

x = 3
x = x + y
print(x)
z = 4
z = z * y
print(z)

// transform to

x = 3
x = x + y
z = 4
z = z * y
print(x)
print(z)
```

```
// register allocation
x = input()
y = input()
print(x + y)
z = input()
print(y + z)
x = input()
print(z + x)
```





Lattice
- meet operator: idempotent, commutative, associative

Partial order for a lattice
- x ^ y = x iff x <= y

Given a function and a partial order
- monotonic: x <= y implies f(x) <= y

(x ^ y) ^ x = (x ^ x) ^ y = x ^ y so (x ^ y) <= x

new <= old, then f(new) <= f(old)

x1 < x2 < x3 < ... xn

```
if (...) {
    x = 3
    y = 4
} else {
    x = 4
    y = 3
}
z = x + y
```

```
// path 1
x = 3
y = 4
z = x + y
// z = 7
```

```
// path 2
x = 4
y = 3
z = x + y
// z = 7
```

// monotonicity
x <= y implies f(x) <= f(y)

how do f(x ^ y) and f(x) ^ f(y) compare?
(x ^ y) <= x and (x ^ y) <= y
f(x ^ y) <= f(x)
f(x ^ y) <= f(y)
f(x ^ y) is a lower bound for f(x) and f(y)
f(x ^ y) <= glb(f(x), f(y)) = f(x) ^ f(y)

// equivalent to monotonicity
f(x ^ y) <= f(x) ^ f(y)

Lattices X, Y, Z
- an element of the product lattice looks like (x, y, z) where x in X, y in Y, z in Z
- (x1, y1, z1) ^ (x2, y2, z2) = (x1 ^ x2, y1 ^ y2, z1 ^ z2)

```
x = 1
z = input()
y = z
while (...) {
	if (y != z) {
		x = 2
	}
	x = 2 - x
	if (x != 1) {
		y = 2
	}
}
print(x) // always 1
```

combining constant propagation with dead code elimination with GVN