Exercises on "Data-Flow Analysis" (UFPE, Recife, Brazil)

1) Undecidability:

Prove that the following problem is *undecidable* (using the "reduction principle"):

- *what are the possible outputs of a program 'P'?*

Let's assume output is done via a special statement (the syntax of which is):

```
STM   ::=   output EXP1 ";"
```

In addition to carrying out the reduction, you need to explain your reasoning. (Hint: it's quite similar to the examples you saw on slides #18+#20 at the lecture.) :-)

2) Control-Flow Diagrams:

Give a *control-flow template* (as the ones on slides #35+#36) for the "**&&**"-construction (aka., *"lazy conjunction"*):

```
EXP   ::=   EXP1 "&&" EXP2
```

You need to *strictly* adhere to the conventions (of drawing)…:

- ***statements*** as rectangles (with flow *in* and *out*);
- ***expressions*** (of type non-boolean) as rectangles (with flow *in* and *out*);
- ***expressions*** (of type boolean) as diamonds (with single flow in and with boolean flow out as two distinct paths, one for "true" and one for "false"); *and*
- ***confluence*** drawn explicitly as circles (collecting multiple flows of control).

3) Control-Flow Graphs:

Draw a *control-flow graph* for the following (silly) program fragment:

```
int N = 5;
int x=input();
int y=input();
for (int i=1; i<N; i++) {
    if (y!=0 && x/y>2) x = x+1;
    else {
        y = y-1;
        while (x>10) x = x/2;
    }
}
output x;
```

(Note: the program isn't supposed to do anything remotely interesting.)

4) Relations and Partial-Orders:

Consider the *subset-of* relation over the set S = $\mathcal{P}$({ **x+1**, **2*y**, **z/3** }) of expressions in a program (written "X ⊆ Y" if X is a subset of Y, in short-hand notation). We'd need such a structure in an analysis that tracks "expressions" (e.g., *"very busy expressions"*-analysis that tracks which expressions have already been computed and haven't changed since). Give:

- its signature;
- the relation (specify its members);
- an example of a member of the relation (both w/ and w/o using short-hand); *and*
- an example of a non-member of the relation (w/ and w/o using short-hand).

Does the set S and relation form a partial-order? (why or why not?)

Draw a Hasse diagram.

5) Greatest-Lower-Bound:

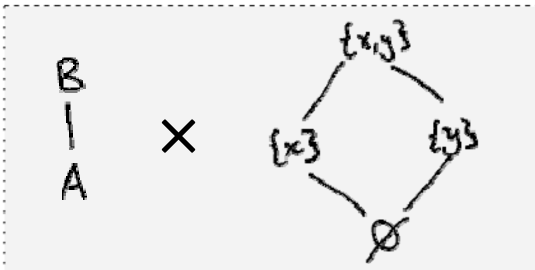
Define the *greatest-lower-bound* (binary operator) on sets '⊔' which is analogous to the *"least-upper-bound"* (binary operator): '⊓' (cf. slide #16 from the 2nd lecture).

Note: it must be: i) *an* lower bound and ii) *the* (i.e., unique) *greatest* lower bound.

Given a lattice **L = (**S**,** ⊆**)**; what do the elements '⊔S' and '⊓S' correspond to?

6) Lattices:

Draw the lattice:



We define the size of a lattice **|L|** as how many elements it has.
In general; how many points will a lattice $\mathbf{L_1}$ **x** $\mathbf{L_2}$ have (assuming $\mathbf{L_1}$ has $|\mathbf{L_1}| = \mathbf{n_1}$ elements and $\mathbf{L_2}$ has $|\mathbf{L_2}| = \mathbf{n_2}$ elements)?

7) Monotone Functions and Fixed-Points:

For each of the 3 recursive equations (over the power-lattice: P({a,b,c}):

i)
```
X = {a,b}
Y = X ∪ Y
```

ii)
```
X = {a,b} ∪ Y
Y = X \ {b}
```

iii)
```
X = {a,b} ∪ Z
Y = {a,c} \ X
Z = X^C
```

Rewrite the equations to bring them onto form: "**x = f(x,y)**" and "**y = g(x,y)**".
Determine whether or not the functions (i.e., '**f**' and '**g**') involved are monotone.

Then, solve the equations that only use monotone functions (i.e., find the [unique] *least fixed point* using the fixed-point theorem).