Ingeniería del Software II

1	2	3	4
B	R	B	M

Parcial #2 – Análisis Estático de Programas

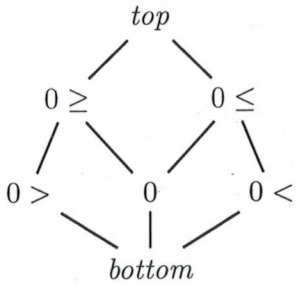
Corrigió Iván

Cada ejercicio se evaluará como **Bien**, **Regular** o **Mal**. Para aprobar el examen es necesario tener al menos 2 ejercicios Bien y al menos 1 ejercicio Regular. Bien = 2,5p, Regular = 1,25p, Mal = 0p.

A

Ejercicio 1

Sea el siguiente reticulado representando los números mayores, iguales y menores a cero:

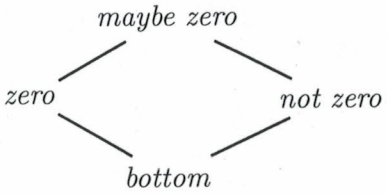


Se desea definir la operación "+" de forma que trate de modelar de la forma más precisa posible la suma de enteros. Definir la función de *transfer* de esta operación para los siguientes casos:

op1	op2	op1 + op2
0	*bottom*	
0	0 >	
0	0	
0	0 <	
0	0 ≤	
0	0 ≥	
0	*top*	
0 ≤	*bottom*	
0 ≤	0 >	
0 ≤	0	
0 ≤	0 <	
0 ≤	0 ≤	
0 ≤	0 ≥	
0 ≤	*top*	

Ejercicio 2

Se tiene el siguiente conjunto de valores abstractos para un análisis *Dataflow may forward* cuyo objetivo es detectar si una variable puede ser (o no) cero durante la ejecución de un programa:



Es decir, cada variable es asignada a uno de los siguientes valores abstractos: *maybe zero*, *zero*, *not zero* o *bottom*. Además, contamos con las funciones de *transfer* para las expresiones:

- "`x = k`" (con k constante):

$$OUT[n](x) = \begin{cases} zero & \text{si } k = 0 \\ not\ zero & \text{si } k \neq 0 \end{cases}$$

- "`x = y + k`" (con k constante):

$$OUT[n](x) = \begin{cases} bottom & \text{si } IN[n](y) = bottom \\ IN[n](y) & \text{si } k = 0 \\ not\ zero & \text{si } IN[n](y) = zero \\ & \wedge\ k \neq 0 \\ maybe\ zero & \text{sino} \end{cases}$$

Dado el siguiente programa:

```
x = 4;
x = x + 1;
y = -1
if x >= 0 {
  y = y + 1;
} else {
  x = x +1;
}
```

a. Construir el CFG del programa.

b. Utilizando el análisis propuesto, calcular el valor abstracto de las variables para los conjunto IN y OUT de cada nodo en el CFG.

Ejercicio 3

Dado el siguiente programa, calcular el grafo de *Points-to* aplicando el algoritmo de *Andersen*.

```
A a = new A(); // call site #1
B b = new B(); // call site #2
a.f = b;
c = a;
c.f = a;
b = a;
```

Ejercicio 4

Sea el sigiuente programa en Java:

```
//@ requires a > 0;
//@ ensures: (a >= b => \result == a)
             && (a < b => \result == b)
             && \result > 0;
int max(int a, int b) {
  int res = b;
  if (a >= b) res = a;
  return res;
}
```

Indique cuáles de las siguientes condiciones podrían ser la *weakest precondition* (WP) del programa. Nota: La instrucción `return E`, asigna la expresión `E` a la variable de especificación `result`.

A. *true*

B. $a > 0 \vee b > 0$

C. $a > 0 \wedge b > 0$

D. $(a \geq b \implies a > 0) \wedge (a < b \implies b > 0)$

¿Es el programa correcto? Si no lo fuera, proponga una nueva *precondición* para que lo sea.