Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI4722 - Lenguajes de Programación III
Abril-Julio 2012

Examen Parcial II

(30 puntos)

Carnet:

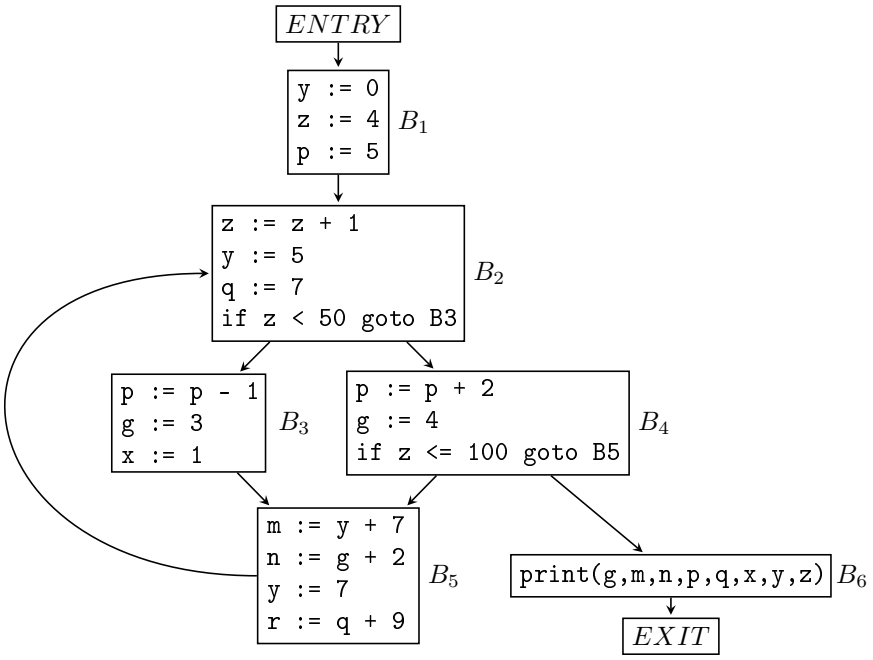
Nombre:

1. **(6 puntos)** Considere el fragmento de código intermedio

```
     x := 42
     y := 6
L1:  z := x + y
     if (x == 17) goto L3
L2:  y = y * 7
     goto L4
     print(y)
L3:  x = x - 1
     if (x > 0) goto L2
L4:  print(x)
     if (z < 1000) goto L1
     return y
```

   Construya el Grafo de Flujo de Control y el Arbol de Dominadores, indicando claramente todos los ciclos naturales, las aristas de retorno y las de retroceso. ¿El grafo resultante es reducible? Demuestre o contradiga.

2. **(6 puntos)** Considere el siguiente Grafo de Flujo



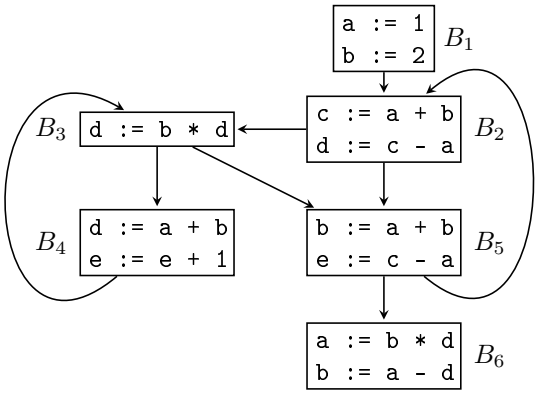
Indique *todas* las expresiones invariantes en ciclos y justifique para cada una de ellas por qué puede o no ser movida si se aplicara el algoritmo de movimiento de código invariante. **No** necesita aplicar el algoritmo.

3. **(6 puntos)** Considere el fragmento de código intermedio

```
     i := 0
L1:  j := 0
     t1 := 160 * i
L2:  t2 := 4 * j
     t3 := t1 + t2
     t4 := a[t3]
     t4 := t4+1
     a[t3] := t4
     j := j + 1
     if j < 40 goto L2
     i = i + 1
     if i < 30 goto L1
```

Aplique el algoritmo de detección de variables de inducción, conduciendo a la reducción de fuerza y posible eliminación de las mismas. Indique claramente las familias de variables identificadas con las tripletas correspondientes, y el código mejorado definitivo.

4. **(6 puntos)** Considere el siguiente Grafo de Flujo



Calcule las expresiones disponibles (*ud chains*), expresiones usadas (*du chains)* y variables vivas, empleando los Algoritmos de Flujo de Datos estudiados en clase. No es necesario el "paso a paso" de los algoritmos.

5. Considere el modelo de lenguaje de programación imperativo **IMP** discutido en el libro de texto "The Formal Semantics of Programming Languages" de Winskel, con la categoría sintáctica de instrucciones

$$I \leftarrow \mathbf{skip} \mid \mathbf{id}{:=}E \mid I_0{;}I_1 \mid \mathbf{if}\, B\, \mathbf{then}\, I_0\, \mathbf{else}\, I_1 \mid \mathbf{while}\, B\, \mathbf{do}\, I_0$$

donde E corresponde a la categoría sintáctica de las expresiones aritméticas y B corresponde a la categoría sintáctica de las expresiones booleanas.

Suponga que se elimina el constructor de instrucciones **while** − **do**, siendo sustituido por el constructor

$$I \leftarrow \mathbf{repeat}\, I_o\, \mathbf{until}\, B$$

a) **(3 puntos)** Defina reglas para Semántica Operacional de la nueva instrucción usando "pasos largos".

b) **(3 puntos)** Defina reglas para Semántica Operacional de la nueva instrucción usando "pasos cortos"

Puede suponer que las reglas de evaluación de expresiones booleanas ya están definidas, tanto en el estilo de "pasos largos" como en el estilo de "pasos cortos". También puede suponer que las reglas para el resto de las instrucciones ya han sido escritas correctamente con ambos estilos. Lo que **no** puede hacer es reutilizar las reglas del **while** − **do** para construir las del **repeat** − **until**.