Ejercicios LTP

Tema 2: Fundamentos de los Lenguajes de Programación

Parte I: Cuestiones

1. De acuerdo a las siguientes reglas BNF:

```
<conditional> ::= IF <cond> THEN <exp> ELSE <exp>;
<cond>        ::= X>0 | X<0
<exp>         ::= X:=X+1 | X:=X-1
```

   Indica si las siguientes sentencias son correctas (puede haber más de una correcta):

   - IF X>0 THEN X:=X+1;
     INCORRECTA. Le falta la parte ELSE.
   - IF X<0 THEN X:=X+1 ELSE X:=X-1;
     CORRECTA.
   - IF X>0 THEN X:=X+1 ELSE X<0;
     INCORRECTA. Detrás de la parte ELSE debe haber una expresión y hay una condición.
   - IF X>0 THEN X:=X-1 ELSE X:=X+1
     INCORRECTA. Le falta el ; final.

2. De acuerdo a las siguientes reglas BNF:

```
<arit> ::= <num> + <num> | <num> - <num>
<expr> ::= <var> = <arit> | <arit> = <var> | <expr> ; <expr>
<num>  ::= 1 | 2 | 3 | 4 | 5
<var>  ::= X | Y | Z
```

   Indica si las siguientes afirmaciones son correctas (puede haber más de una correcta):

   - 1+1 es una expresión <arit>.
     CORRECTA.
   - 1+2-3 es una expresión <arit>.
     INCORRECTA.
   - 1+2=X es una expresión <expr>.
     CORRECTA.
   - Z=2+3;Y=1-4 es una expresión <expr>.
     CORRECTA.

3. Analiza si el siguiente programa escrito en C-Minus es válido con respecto a la gramática incluida en el Apéndice, que describe la sintaxis de un pequeño subconjunto de C (llamado C-Minus):

```
long factorial(int n)
{
 int c = 2;
 long result = 1;

 while (c <= n)
 { result = result * c;
   c++
 }
 return result;
}
```

Respuesta (sí/no): No

Respuesta: **NO**
Justificación: De acuerdo con la gramática BNF de C-minus del apéndice, las declaraciones de variables tienen la forma siguiente:

var-declaration → *type-specifier* **ID ;** | *type-specifier* **ID** [**NUM**]**;**

Por lo que la sintaxis no permite expresiones de inicialización en la declaración.
Otros errores son:

- El uso del tipo `long` no se permite.
- El operador posincremento no está permitido.
- A cada instrucción debe seguirle un punto y coma.

4. Da las reglas semánticas para la evaluación de la operación booleana de disyunción.

$$\frac{\langle b_0, s\rangle \Rightarrow \mathit{true}}{\langle b_0 \vee b_1, s\rangle \Rightarrow \mathit{true}} \qquad \frac{\langle b_0, s\rangle \Rightarrow \mathit{false} \qquad \langle b_1, s\rangle \Rightarrow \beta}{\langle b_0 \vee b_1, s\rangle \Rightarrow \beta}$$

5. Dado el siguiente código P:

```
X:=5;
Y:=X
```

(a) Desarrolla la traza de ejecución (mostrando los cómputos intermedios) siguiendo la semántica *small-step* con el estado inicial $s_I = \{\}$).

La traza consta de tres pasos (de la configuración **(1)** a la configuración **(4)**:

(1) $\langle(\mathtt{X} := 5; \mathtt{Y} := \mathtt{X}), \{\}\rangle \rightarrow$
$\quad\quad\langle\mathtt{X} := 5, \{\}\rangle \rightarrow$
$\quad\quad\quad\langle 5, \{\}\rangle \Rightarrow 5$
$\quad\quad\langle skip, \{X \mapsto 5\}\rangle$
(2) $\langle(\mathtt{skip}; \mathtt{Y} := \mathtt{X}), \{X \mapsto 5\}\rangle \rightarrow$
(3) $\langle\mathtt{Y} := \mathtt{X}, \{X \mapsto 5\}\rangle \rightarrow$
$\quad\quad\langle\mathtt{X}, \{X \mapsto 5\}\rangle \Rightarrow 5$
(4) $\langle\mathtt{skip}, \{X \mapsto 5, Y \mapsto 5\}\rangle$

Los cómputos intermedios (que aparecen entre pasos identados) corresponden a las pruebas de las condiciones en la premisa de la semántica operacional de paso pequeño. Por ejemplo, la tercera línea ($\langle 5, \{\}\rangle \Rightarrow 5$)) corresponde a la premisa $\langle a, s\rangle \Rightarrow n$ que realiza la evaluación de la expresión artitmética a en la aplicación de la regla de asignación en la segunda y cuarta líneas:

$$\frac{\langle a, s\rangle \Rightarrow n}{\langle x{:=}a, s\rangle \rightarrow \langle skip, s[x \mapsto n]\rangle}$$

(b) Calcula la semántica *big-step*, mostrando los cómputos intermedios, con el estado inicial vacío.

$\langle\mathtt{X} := 5; \mathtt{Y} := \mathtt{X}, \{\}\rangle$
$\quad\quad\langle\mathtt{X} := 5, \{\}\rangle$
$\quad\quad\quad\langle 5, \{\}\rangle \Rightarrow 5$
$\quad\quad\Downarrow \{X \mapsto 5\}$
$\quad\quad\langle\mathtt{Y} := \mathtt{X}, \{X \mapsto 5\}\rangle$
$\quad\quad\quad\langle\mathtt{X}, \{X \mapsto 5\}\rangle \Rightarrow 5$
$\quad\quad\Downarrow \{X \mapsto 5, Y \mapsto 5\}$
$\Downarrow \{X \mapsto 5, Y \mapsto 5\}$

6. Dada la siguiente configuración, desarrolla la evaluación de la expresión aritmética aplicando las reglas correspondientes:

$$\langle\mathtt{X} + 3, \{X \mapsto 2\}\rangle \Rightarrow \ldots$$

$\langle\mathtt{X} + 3, \{X \mapsto 2\}\rangle \Rightarrow$
$\quad\quad\langle\mathtt{X}, \{X \mapsto 2\}\rangle \Rightarrow 2$
$\quad\quad\langle 3, \{X \mapsto 2\}\rangle \Rightarrow 3$
5

7. Dada la siguiente configuración, desarrolla la evaluación de la expresión booleana aplicando las reglas correspondientes:

$$\langle \mathtt{X} + \mathtt{3} \leq \mathtt{Y}, \{X \mapsto 2, Y \mapsto 0\}\rangle \Rightarrow \ldots$$

$\langle \mathtt{X} + \mathtt{3} \leq \mathtt{Y}, \{X \mapsto 2, Y \mapsto 0\}\rangle \Rightarrow$
$\quad\langle \mathtt{X} + \mathtt{3}, \{X \mapsto 2, Y \mapsto 0\}\rangle \Rightarrow$
$\quad\quad\langle \mathtt{X}, \{X \mapsto 2, Y \mapsto 0\}\rangle \Rightarrow 2$
$\quad\quad\langle \mathtt{3}, \{X \mapsto 2, Y \mapsto 0\}\rangle \Rightarrow 3$
$\quad 5$
$\quad\langle \mathtt{Y}, \{X \mapsto 2, Y \mapsto 0\}\rangle \Rightarrow 0$
$\quad 5 \leq 0$ is false
false

8. Dado el siguiente código P:

```
X:=5;
if X>3 then X:= X-1 else Y:=X
```

(a) Desarrolla la traza de ejecución (mostrando los cómputos intermedios) siguiendo la semántica *small-step* con el estado inicial $\{X \mapsto 2\}$.

(1) $\langle (\mathtt{X} := \mathtt{5}; \mathtt{if}\ \mathtt{X} > \mathtt{3}\ \mathtt{then}\ \mathtt{X} := \mathtt{X} - \mathtt{1}\ \mathtt{else}\ \mathtt{Y} := \mathtt{X}), \{X \mapsto 2\}\rangle \rightarrow$
$\quad\langle \mathtt{X} := \mathtt{5}, \{X \mapsto 2\}\rangle \rightarrow$
$\quad\quad\langle 5, \{X \mapsto 2\}\rangle \Rightarrow 5$
$\quad\langle \mathtt{skip}, \{X \mapsto 5\}\rangle$
(2) $\langle (\mathtt{skip}; \mathtt{if}\ \mathtt{X} > \mathtt{3}\ \mathtt{then}\ \mathtt{X} := \mathtt{X} - \mathtt{1}\ \mathtt{else}\ \mathtt{Y} := \mathtt{X}), \{X \mapsto 5\}\rangle \rightarrow$
(3) $\langle \mathtt{if}\ \mathtt{X} > \mathtt{3}\ \mathtt{then}\ \mathtt{X} := \mathtt{X} - \mathtt{1}\ \mathtt{else}\ \mathtt{Y} := \mathtt{X}, \{X \mapsto 5\}\rangle \rightarrow$
$\quad\langle X > 3, \{X \mapsto 5\}\rangle \Rightarrow$
$\quad\quad\langle X, \{X \mapsto 5\}\rangle \Rightarrow 5$
$\quad\quad\langle 3, \{X \mapsto 5\}\rangle \Rightarrow 3$
$\quad\quad 5 > 3$ is true
$\quad$ *true*
(4) $\langle \mathtt{X} := \mathtt{X} - \mathtt{1}, \{X \mapsto 5\}\rangle \rightarrow$
$\quad\langle \mathtt{X} - \mathtt{1}, \{X \mapsto 5\}\rangle \Rightarrow$
$\quad\quad\langle X, \{X \mapsto 5\}\rangle \Rightarrow 5$
$\quad\quad\langle 1, \{X \mapsto 5\}\rangle \Rightarrow 1$
$\quad\quad 5 - 1 = 4$ is computed
$\quad 4$
(5) $\langle \mathtt{skip}, \{X \mapsto 4\}\rangle$

(b) Calcula la semántica *big-step*, mostrando los cómputos intermedios, con el estado inicial $\{X \mapsto 2\}$.

$\langle (\mathtt{X} := \mathtt{5}; \mathtt{if}\ \mathtt{X} > \mathtt{3}\ \mathtt{then}\ \mathtt{X} := \mathtt{X} - \mathtt{1}\ \mathtt{else}\ \mathtt{Y} := \mathtt{X}), \{X \mapsto 2\}\rangle$

$\quad\langle \mathtt{X} := \mathtt{5}, \{X \mapsto 2\}\rangle$

$\quad\quad\langle 5, \{X \mapsto 2\}\rangle \Rightarrow 5$

$\quad\Downarrow \{X \mapsto 5\}$

$\quad\langle \mathtt{if}\ \mathtt{X} > \mathtt{3}\ \mathtt{then}\ \mathtt{X} := \mathtt{X} - \mathtt{1}\ \mathtt{else}\ \mathtt{Y} := \mathtt{X}, \{X \mapsto 5\}\rangle$

$\quad\quad\langle \mathtt{X} > \mathtt{3}, \{X \mapsto 5\}\rangle \Rightarrow$

$\quad\quad\quad\langle \mathtt{X}, \{X \mapsto 5\}\rangle \Rightarrow 5$

$\quad\quad\quad\langle \mathtt{3}, \{X \mapsto 5\}\rangle \Rightarrow 3$

$\quad\quad\quad 5 > 3$ is true

$\quad\quad$ *true*

$\quad\quad\langle \mathtt{X} := \mathtt{X} - \mathtt{1}, \{X \mapsto 5\}\rangle$

$\quad\quad\quad\langle \mathtt{X} - \mathtt{1}, \{X \mapsto 5\}\rangle \Rightarrow$

$\quad\quad\quad\quad\langle \mathtt{X}, \{X \mapsto 5\}\rangle \Rightarrow 5$

$\quad\quad\quad\quad\langle \mathtt{1}, \{X \mapsto 5\}\rangle \Rightarrow 1$

$\quad\quad\quad\quad 5 - 1 = 4$ is computed

$\quad\quad\quad 4$

$\quad\quad\Downarrow \{X \mapsto 4\}$

$\quad\Downarrow \{X \mapsto 4\}$

$\Downarrow \{X \mapsto 4\}$

9. Queremos extender el lenguaje SIMP visto en las transparencias con una nueva instrucción *repeat* cuya sintaxis es

```
repeat i until b
```

El comportamiento de esta instrucción es el siguiente: debe ejecutarse la instrucción (posiblemente compuesta) `i` hasta que se satisfaga la condición `b`, en cuyo momento se detendrá la ejecución del *repeat*.

(a) Da la(s) regla(s) semántica(s) siguiendo el estilo *small-step* par la instrucción *repeat*

Solución A: Traducción en un bucle `while`:

$$\frac{}{\langle \mathtt{repeat\ i\ until\ b}, s\rangle \rightarrow \langle \mathtt{i}; \mathtt{while}\ \neg\mathtt{b}\ \mathtt{do}\ \mathtt{i}, s\rangle}$$

Solución B: Definición directa:

$$\frac{\langle \mathtt{b}, s\rangle \Rightarrow \mathit{true}}{\langle \mathtt{repeat\ skip\ until\ b}, s\rangle \rightarrow \langle \mathtt{skip}, s\rangle}$$

$$\frac{\langle \mathtt{b}, s\rangle \Rightarrow \mathit{false}}{\langle \mathtt{repeat\ skip\ until\ b}, s\rangle \rightarrow \langle \mathtt{repeat\ skip\ until\ b}, s\rangle}$$

$$\frac{\langle \mathtt{i}, s\rangle \rightarrow \langle \mathtt{i}', s'\rangle}{\langle \mathtt{repeat\ i\ until\ b}, s\rangle \rightarrow \langle \mathtt{i}'; \mathtt{if\ b\ then\ skip\ else\ (repeat\ i\ until\ b)}, s'\rangle}\quad \text{if } \mathtt{i} \neq \mathtt{skip}$$

(b) Da la(s) regla(s) semántica(s) siguiendo el estilo *big-step* par la instrucción *repeat*

$$\frac{\langle i,\ e\rangle\ \Downarrow\ e' \qquad \langle b,\ e'\rangle \Rightarrow true}{\langle \texttt{repeat}\ i\ \texttt{until}\ b,\ e\rangle\ \ \Downarrow\ \ e'}$$

$$\frac{\langle i,\ e\rangle\ \Downarrow\ e' \qquad \langle b,\ e'\rangle \Rightarrow false \qquad \langle \texttt{repeat}\ i\ \texttt{until}\ b,\ e'\rangle\ \Downarrow\ \ e''}{\langle \texttt{repeat}\ i\ \texttt{until}\ b,\ e\rangle\ \ \Downarrow\ \ e''}$$

10. Dado el siguiente código P:

```
X:=4;
while X>3 do X:= X-1
```

(a) Desarrolla la traza de ejecución (mostrando los cómputos intermedios) siguiendo la semántica *small-step* con el estado inicial vacío.

(1) $\langle(\mathtt{X} := \mathtt{4}; \mathtt{while}\ \mathtt{X} > \mathtt{3}\ \mathtt{do}\ \mathtt{X} := \mathtt{X} - \mathtt{1}), \{\}\rangle \rightarrow$

$\quad\langle \mathtt{X} := \mathtt{4}, \{\}\rangle \rightarrow$

$\quad\quad\langle \mathtt{4}, \{\}\rangle \Rightarrow 4$

$\quad\langle \mathtt{skip}, \{X \mapsto 4\}\rangle$

(2) $\langle(\mathtt{skip}; \mathtt{while}\ \mathtt{X} > \mathtt{3}\ \mathtt{do}\ \mathtt{X} := \mathtt{X} - \mathtt{1}), \{X \mapsto 4\}\rangle \rightarrow$

(3) $\langle \mathtt{while}\ \mathtt{X} > \mathtt{3}\ \mathtt{do}\ \mathtt{X} := \mathtt{X} - \mathtt{1}, \{X \mapsto 4\}\rangle \rightarrow$

$\quad\langle \mathtt{X} > \mathtt{3}, \{X \mapsto 4\}\rangle \Rightarrow$

$\quad\quad\langle \mathtt{X}, \{X \mapsto 4\}\rangle \Rightarrow 4$

$\quad\quad\langle \mathtt{3}, \{X \mapsto 4\}\rangle \Rightarrow 3$

$\quad\quad 4 > 3$ holds

$\quad$ *true*

(4) $\langle(\mathtt{X} := \mathtt{X} - \mathtt{1}; \mathtt{while}\ \mathtt{X} > \mathtt{3}\ \mathtt{do}\ \mathtt{X} := \mathtt{X} - \mathtt{1}), \{X \mapsto 4\}\rangle \rightarrow$

$\quad\langle \mathtt{X} := \mathtt{X} - \mathtt{1}, \{X \mapsto 4\}\rangle \rightarrow$

$\quad\quad\langle \mathtt{X} - \mathtt{1}, \{X \mapsto 4\}\rangle \Rightarrow$

$\quad\quad\quad\langle \mathtt{X}, \{X \mapsto 4\}\rangle \Rightarrow 4$

$\quad\quad\quad\langle \mathtt{1}, \{X \mapsto 4\}\rangle \Rightarrow 1$

$\quad\quad\quad 4 - 1 = 3$ is obtained

$\quad\quad 3$

$\quad\langle \mathtt{skip}, \{X \mapsto 3\}\rangle$

(5) $\langle(\mathtt{skip}; \mathtt{while}\ \mathtt{X} > \mathtt{3}\ \mathtt{do}\ \mathtt{X} := \mathtt{X} - \mathtt{1}), \{X \mapsto 3\}\rangle \rightarrow$

(6) $\langle \mathtt{while}\ \mathtt{X} > \mathtt{3}\ \mathtt{do}\ \mathtt{X} := \mathtt{X} - \mathtt{1}, \{X \mapsto 3\}\rangle \rightarrow$

$\quad\langle \mathtt{X} > \mathtt{3}, \{X \mapsto 3\}\rangle \Rightarrow$

$\quad\quad\langle \mathtt{X}, \{X \mapsto 3\}\rangle \Rightarrow 3$

$\quad\quad\langle \mathtt{3}, \{X \mapsto 3\}\rangle \Rightarrow 3$

$\quad\quad 3 > 3$ is false

$\quad$ *false*

(7) $\langle \mathtt{skip}, \{X \mapsto 3\}\rangle$

(b) Calcula la semántica *big-step*, mostrando los cómputos intermedios, con el estado inicial $s = \{\}$.

$\langle(\mathtt{X} := \mathtt{4}; \mathtt{while}\ \mathtt{X} > \mathtt{3}\ \mathtt{do}\ \mathtt{X} := \mathtt{X} - \mathtt{1}), \{\}\rangle$
$\quad\langle \mathtt{X} := \mathtt{4}, \{\}\rangle$
$\quad\quad\langle \mathtt{4}, \{\}\rangle \Rightarrow 4$
$\quad\Downarrow \{X \mapsto 4\}$
$\quad\langle \mathtt{while}\ \mathtt{X} > \mathtt{3}\ \mathtt{do}\ \mathtt{X} := \mathtt{X} - \mathtt{1}, \{X \mapsto 4\}\rangle$
$\quad\quad\langle \mathtt{X} > \mathtt{3}, \{X \mapsto 4\}\rangle \Rightarrow$
$\quad\quad\quad\langle \mathtt{X}, \{X \mapsto 4\}\rangle \Rightarrow 4$
$\quad\quad\quad\langle \mathtt{3}, \{X \mapsto 4\}\rangle \Rightarrow 3$
$\quad\quad\quad 4 > 3$ es true
$\quad\quad$ *true*
$\quad\quad\langle(\mathtt{X} := \mathtt{X} - \mathtt{1}; \mathtt{while}\ \mathtt{X} > \mathtt{3}\ \mathtt{do}\ \mathtt{X} := \mathtt{X} - \mathtt{1}), \{X \mapsto 4\}\rangle$
$\quad\quad\quad\langle \mathtt{X} := \mathtt{X} - \mathtt{1}, \{X \mapsto 4\}\rangle$
$\quad\quad\quad\quad\langle \mathtt{X} - \mathtt{1}, \{X \mapsto 4\}\rangle \Rightarrow$
$\quad\quad\quad\quad\quad\langle \mathtt{X}, \{X \mapsto 4\}\rangle \Rightarrow 4$
$\quad\quad\quad\quad\quad\langle \mathtt{1}, \{X \mapsto 4\}\rangle \Rightarrow 1$
$\quad\quad\quad\quad\quad 4 - 1 = 3$ es computado
$\quad\quad\quad\quad 3$
$\quad\quad\quad\Downarrow \{X \mapsto 3\}$
$\quad\quad\langle \mathtt{while}\ \mathtt{X} > \mathtt{3}\ \mathtt{do}\ \mathtt{X} := \mathtt{X} - \mathtt{1}, \{X \mapsto 3\}\rangle$
$\quad\quad\quad\langle \mathtt{X} > \mathtt{3}, \{X \mapsto 3\}\rangle \Rightarrow$
$\quad\quad\quad\quad\langle \mathtt{X}, \{X \mapsto 3\}\rangle \Rightarrow 3$
$\quad\quad\quad\quad\langle \mathtt{3}, \{X \mapsto 3\}\rangle \Rightarrow 3$
$\quad\quad\quad\quad 3 > 3$ es false
$\quad\quad\quad$ *false*
$\quad\quad\Downarrow \{X \mapsto 3\}$
$\quad\Downarrow \{X \mapsto 3\}$
$\Downarrow \{X \mapsto 3\}$

11. Queremos extender el lenguaje SIMP visto en las transparencias con una nueva instrucción *for* cuya sintaxis es

```
for V:=a0 to a1 do i
```

El comportamiento de esta instrucción es el siguiente: la variable `V` (el "contador") va tomando valores, en orden creciente, desde `a0` hasta `a1` (ambos inclusive) y, tras cada asignación, se ejecuta la instrucción `i`.

(a) Da la(s) regla(s) semántica(s) siguiendo el estilo *small-step* par la instrucción *for*

Solución A: Definición indirecta, mediante transformación a la instrucción *while*:

$$\frac{}{\langle \texttt{for V := a0 to a1 do i}, s\rangle \rightarrow \langle (\texttt{V := a0; while V <= a1 do (i; V := V + 1)}), s\rangle}$$

Solución B (consta de dos reglas):

$$\frac{\langle \texttt{a0} > \texttt{a1}, s\rangle \Rightarrow \textit{true}}{\langle \texttt{for V := a0 to a1 do i}, s\rangle \rightarrow \langle \texttt{skip}, s\rangle}$$

$$\frac{\langle \texttt{a0} > \texttt{a1}, s\rangle \Rightarrow \textit{false}}{\langle \texttt{for V := a0 to a1 do i}, s\rangle \rightarrow \langle (\texttt{V := a0; i; for V := V + 1 to a1 do i}), s\rangle}$$

Nótese en la solución B la necesidad de ejecutar la asignación del valor `a0` a la variable de control `V` antes de ejecutar `i` ya que podría ser usada durante la ejecución de `i`. También, la 'inicialización' de `V` a `V+1` en la segunda llamada del bucle implementa el incremento del contador a pesar de que `V` pueda cambiar durante la ejecución de `i`.

(b) Da la(s) regla(s) semántica(s) siguiendo el estilo *big-step* par la instrucción *for*

$$\frac{\langle \texttt{a0} > \texttt{a1}, s\rangle \Rightarrow \textit{true}}{\langle \texttt{for V := a0 to a1 do i}, s\rangle \Downarrow s}$$

$$\frac{\langle \texttt{a0} > \texttt{a1}, s\rangle \Rightarrow \textit{false} \quad \langle \texttt{V := a0}, s\rangle \Downarrow s' \quad \langle \texttt{i}, s'\rangle \Downarrow s'' \quad \langle \texttt{for V := V + 1 to a1 do i}, s''\rangle \Downarrow s'''}{\langle \texttt{for V := a0 to a1 do i}, s\rangle \Downarrow s'''}$$

12. Queremos extender el lenguaje SIMP visto en las transparencias con una nueva instrucción *times* cuya sintaxis es

```
do n times i
```

El comportamiento de esta instrucción es el siguiente: se ejecuta la instrucción `i` tantas veces como indique el número natural `n` (si `n=0` la instrucción `i` no se ejecuta).

(a) Da la(s) regla(s) semántica(s) siguiendo el estilo *small-step* par la instrucción *times*

Solución A:

$$\frac{\langle \texttt{n} > \texttt{0}, s\rangle \Rightarrow \textit{false}}{\langle \texttt{do n times i}, s\rangle \rightarrow \langle \texttt{skip}, s\rangle}$$

$$\frac{\langle \texttt{n} > \texttt{0}, s\rangle \Rightarrow \textit{true}}{\langle \texttt{do n times i}, s\rangle \rightarrow \langle (\texttt{i; do (n - 1) times i}), s\rangle}$$

Solución B:

$$\frac{}{\langle \texttt{do n times i}, s\rangle \rightarrow \langle \texttt{while n > 0 do (i; n := n - 1)}, s\rangle}$$

(b) Da la(s) regla(s) semántica(s) siguiendo el estilo *big-step* par la instrucción *times*

$$\frac{\langle \mathtt{n} > \mathtt{0}, s\rangle \Rightarrow \mathit{false}}{\langle \mathtt{do\ n\ times\ i}, s\rangle \Downarrow s}$$

$$\frac{\langle \mathtt{n} > \mathtt{0}, s\rangle \Rightarrow \mathit{true} \qquad \langle \mathtt{i}, s\rangle \Downarrow s' \qquad \langle \mathtt{do\ (n-1)\ times\ i}, s'\rangle \Downarrow s''}{\langle \mathtt{do\ n\ times\ i}, s\rangle \Downarrow s''}$$

13. Dado el siguiente código S donde se calcula el máximo de dos números:

```
if X>Y then max:=X else max:=Y
```

(a) Desarrolla la traza de ejecución (mostrando los cómputos intermedios) siguiendo la semántica *small-step* con el estado inicial $\{X \mapsto 3, Y \mapsto 5\}$.

(1) $\langle \mathtt{if\ X} > \mathtt{Y\ then\ max} := \mathtt{X\ else\ max} := \mathtt{Y}, \{X \mapsto 3, Y \mapsto 5\}\rangle \rightarrow$
$\langle \mathtt{X} > \mathtt{Y}, \{X \mapsto 3, Y \mapsto 5\}\rangle \Rightarrow$
$\langle \mathtt{X}, \{X \mapsto 3, Y \mapsto 5\}\rangle \Rightarrow 3$
$\langle \mathtt{Y}, \{X \mapsto 3, Y \mapsto 5\}\rangle \Rightarrow 5$
$3 > 5$ es false
false
(2) $\langle \mathtt{max} := \mathtt{Y}, \{X \mapsto 3, Y \mapsto 5\}\rangle \rightarrow$
$\langle \mathtt{Y}, \{X \mapsto 3, Y \mapsto 5\}\rangle \Rightarrow 5$
(3) $\langle \mathtt{skip}, \{X \mapsto 3, Y \mapsto 5, \mathit{max} \mapsto 5\}\rangle$

(b) Calcula la semántica *big-step*, mostrando los cómputos intermedios, con el estado inicial $\{X \mapsto 3, Y \mapsto 5\}$.

$\langle \mathtt{if\ X} > \mathtt{Y\ then\ max} := \mathtt{X\ else\ max} := \mathtt{Y}, \{X \mapsto 3, Y \mapsto 5\}\rangle$
$\langle \mathtt{X} > \mathtt{Y}, \{X \mapsto 3, Y \mapsto 5\}\rangle \Rightarrow$
$\langle \mathtt{X}, \{X \mapsto 3, Y \mapsto 5\}\rangle \Rightarrow 3$
$\langle \mathtt{Y}, \{X \mapsto 3, Y \mapsto 5\}\rangle \Rightarrow 5$
$3 > 5$ es false
false
$\langle \mathtt{max} := \mathtt{Y}, \{X \mapsto 3, Y \mapsto 5\}\rangle$
$\langle \mathtt{Y}, \{X \mapsto 3, Y \mapsto 5\}\rangle \Rightarrow 5$
$\Downarrow \{X \mapsto 3, Y \mapsto 5, \mathit{max} \mapsto 5\}$
$\Downarrow \{X \mapsto 3, Y \mapsto 5, \mathit{max} \mapsto 5\}$

14. Queremos extender el lenguaje SIMP visto en las transparencias con un nuevo operador de asignación múltiple cuya sintaxis es

$\mathtt{x}_1\mathtt{,x}_2\mathtt{,...x}_n \mathtt{\ :=\ a}_1\mathtt{,a}_2\mathtt{,...a}_n$

donde

- las $\mathtt{x}_i$ son distintas entre sí,
- las $\mathtt{a}_i$ son expresiones

y su comportamiento debe ser el siguiente (ver [Gries81], página 121):

- primero se evalúan las expresiones $\mathtt{a}_1, \mathtt{a}_2, \ldots \mathtt{a}_n$ (en cualquier orden), obteniéndose los valores $\mathtt{v}_1, \mathtt{v}_2, \ldots \mathtt{v}_n$ respectivamente;
- para cada i, se asigna a $\mathtt{x}_i$ el valor $\mathtt{v}_i$.

(a) Da la(s) regla(s) semántica(s) siguiendo el estilo *small-step* par la instrucción *asignación-múltiple*

$$\frac{\langle \mathtt{a_1}, s\rangle \Rightarrow v_1 \qquad \langle \mathtt{a_2}, s\rangle \Rightarrow v_2 \qquad \cdots \qquad \langle \mathtt{a_n}, s\rangle \Rightarrow v_n}{\langle \mathtt{x_1}, \mathtt{x_2}, \ldots \mathtt{x_n} := \mathtt{a_1}, \mathtt{a_2}, \ldots \mathtt{a_n}, s\rangle \to \langle \mathtt{skip}, s[x_1 \mapsto v_1, x_2 \mapsto v_2, \ldots x_n \mapsto v_n]\rangle}$$

(b) Da la(s) regla(s) semántica(s) siguiendo el estilo *big-step* par la instrucción *asignación-múltiple*

$$\frac{\langle \mathtt{a_1}, s\rangle \Rightarrow v_1 \qquad \langle \mathtt{a_2}, s\rangle \Rightarrow v_2 \qquad \cdots \qquad \langle \mathtt{a_n}, s\rangle \Rightarrow v_n}{\langle \mathtt{x_1}, \mathtt{x_2}, \ldots \mathtt{x_n} := \mathtt{a_1}, \mathtt{a_2}, \ldots \mathtt{a_n}, s\rangle \Downarrow s[x_1 \mapsto v_1, x_2 \mapsto v_2, \ldots x_n \mapsto v_n]}$$

15. Dado el siguiente programa S:

```
t:=x;
x:=y;
y:=t;
```

Desarrolla la traza de la ejecución a partir del estado $\{x \mapsto 2, y \mapsto 5\}$ usando la semántica operacional de paso pequeño.

(1) $\langle (\mathtt{t} := \mathtt{x}; \mathtt{x} := \mathtt{y}; \mathtt{y} := \mathtt{t}), \{x \mapsto 2, y \mapsto 5\}\rangle \to$
$\quad\langle \mathtt{t} := \mathtt{x}, \{x \mapsto 2, y \mapsto 5\}\rangle \to$
$\quad\quad\langle \mathtt{x}, \{x \mapsto 2, y \mapsto 5\}\rangle \Rightarrow 2$
$\quad\langle \mathtt{skip}, \{x \mapsto 2, y \mapsto 5, t \mapsto 2\}\rangle$
(2) $\langle (\mathtt{skip}; \mathtt{x} := \mathtt{y}; \mathtt{y} := \mathtt{t}), \{x \mapsto 2, y \mapsto 5, t \mapsto 2\}\rangle \to$
(3) $\langle (\mathtt{x} := \mathtt{y}; \mathtt{y} := \mathtt{t}), \{x \mapsto 2, y \mapsto 5, t \mapsto 2\}\rangle \to$
$\quad\langle \mathtt{x} := \mathtt{y}, \{x \mapsto 2, y \mapsto 5, t \mapsto 2\}\rangle \to$
$\quad\quad\langle \mathtt{y}, \{x \mapsto 2, y \mapsto 5, t \mapsto 2\}\rangle \Rightarrow 5$
$\quad\langle \mathtt{skip}, \{x \mapsto 5, y \mapsto 5, t \mapsto 2\}\rangle$
(4) $\langle (\mathtt{skip}; \mathtt{y} := \mathtt{t}), \{x \mapsto 5, y \mapsto 5, t \mapsto 2\}\rangle \to$
(5) $\langle \mathtt{y} := \mathtt{t}, \{x \mapsto 5, y \mapsto 5, t \mapsto 2\}\rangle \to$
$\quad\langle \mathtt{t}, \{x \mapsto 5, y \mapsto 5, t \mapsto 2\}\rangle \Rightarrow 2$
(6) $\langle \mathtt{skip}, \{x \mapsto 5, y \mapsto 2, t \mapsto 2\}\rangle$

16. Dado el siguiente fragmento de código P:

```
x:=x+1;
y:=y+x;
x:=x+1;
```

usando la semántica operacional de paso pequeño, construye la traza de ejecución a partir del estado inicial $\{x \mapsto 3, y \mapsto 7\}$.

(1) $\langle \mathtt{x} := \mathtt{x}+\mathtt{1}; \mathtt{y} := \mathtt{y}+\mathtt{x}; \mathtt{x} := \mathtt{x}+\mathtt{1}, \{x \mapsto 3, y \mapsto 7\}\rangle \rightarrow$
  $\langle \mathtt{x} := \mathtt{x}+\mathtt{1}, \{x \mapsto 3, y \mapsto 7\}\rangle \rightarrow$
    $\langle \mathtt{x}+\mathtt{1}, \{x \mapsto 3, y \mapsto 7\}\rangle \Rightarrow$
      $\langle \mathtt{x}, \{x \mapsto 3, y \mapsto 7\}\rangle \Rightarrow 3$
      $\langle \mathtt{1}, \{x \mapsto 3, y \mapsto 7\}\rangle \Rightarrow 1$
      $3+1=4$ es computado
    4
  $\langle \mathtt{skip}, \{x \mapsto 4, y \mapsto 7\}\rangle$
(2) $\langle \mathtt{skip}; \mathtt{y} := \mathtt{y}+\mathtt{x}; \mathtt{x} := \mathtt{x}+\mathtt{1}, \{x \mapsto 4, y \mapsto 7\}\rangle \rightarrow$
(3) $\langle \mathtt{y} := \mathtt{y}+\mathtt{x}; \mathtt{x} := \mathtt{x}+\mathtt{1}, \{x \mapsto 4, y \mapsto 7\}\rangle$
  $\langle \mathtt{y} := \mathtt{y}+\mathtt{x}, \{x \mapsto 4, y \mapsto 7\}\rangle \rightarrow$
    $\langle \mathtt{y}+\mathtt{x}, \{x \mapsto 4, y \mapsto 7\}\rangle \Rightarrow$
      $\langle \mathtt{y}, \{x \mapsto 4, y \mapsto 7\}\rangle \Rightarrow 7$
      $\langle \mathtt{x}, \{x \mapsto 4, y \mapsto 7\}\rangle \Rightarrow 4$
      $7+4=11$ es computado
    11
  $\langle \mathtt{skip}, \{x \mapsto 4, y \mapsto 11\}\rangle$
(4) $\langle \mathtt{skip}; \mathtt{x} := \mathtt{x}+\mathtt{1}, \{x \mapsto 4, y \mapsto 11\}\rangle \rightarrow$
(5) $\langle \mathtt{x} := \mathtt{x}+\mathtt{1}, \{x \mapsto 4, y \mapsto 11\}\rangle \rightarrow$
  $\langle \mathtt{x}+\mathtt{1}, \{x \mapsto 4, y \mapsto 11\}\rangle \Rightarrow$
    $\langle \mathtt{x}, \{x \mapsto 4, y \mapsto 11\}\rangle \Rightarrow 4$
    $\langle \mathtt{1}, \{x \mapsto 4, y \mapsto 11\}\rangle \Rightarrow 1$
    $4+1=5$ es computado
  5
(5) $\langle \mathtt{skip}, \{x \mapsto 5, y \mapsto 11\}\rangle$

17. Considera el siguiente código en C que devuelve el máximo de dos números:

```
int maximo (int x, int y)
{
if (x>y)
return x ;
else
return y ;
} ;
```

(a) Escribe el cuerpo de la función `maximo` en la sintaxis del lenguaje SIMP (en SIMP no hay subprogramas).

Traducimos el cuerpo a la sintaxis SIMP

```
if (x>y)
 then m := x
 else m := y
```

(b) Construye las trazas de la ejecución de la llamada `maximo(3,5)` (es decir, partiendo del estado inicial $\{x \mapsto 3, y \mapsto 5\}$), usando las semánticas de paso pequeño y grande.

Small-step

(1) $\langle \texttt{if (x > y) then m := x else m := y}, \{x \mapsto 3, y \mapsto 5\}\rangle \rightarrow$

$\quad\quad \langle \texttt{x > y}, \{x \mapsto 3, y \mapsto 5\}\rangle \Rightarrow$

$\quad\quad\quad\quad \langle \texttt{x}, \{x \mapsto 3, y \mapsto 5\}\rangle \Rightarrow 3$

$\quad\quad\quad\quad \langle \texttt{y}, \{x \mapsto 3, y \mapsto 5\}\rangle \Rightarrow 5$

$\quad\quad\quad\quad 3 > 5$ es false

$\quad\quad$ *false*

(2) $\langle \texttt{m := y}, \{x \mapsto 3, y \mapsto 5\}\rangle \rightarrow$

$\quad\quad \langle \texttt{y}, \{x \mapsto 3, y \mapsto 5\}\rangle \Rightarrow 5$

(3) $\langle \texttt{skip}, \{x \mapsto 3, y \mapsto 5, m \mapsto 5\}\rangle \Rightarrow 5$

Big-step

$\langle \texttt{if (x > y) then m := x else m := y}, \{x \mapsto 3, y \mapsto 5\}\rangle$

$\quad \langle \texttt{x > y}, \{x \mapsto 3, y \mapsto 5\}\rangle \Rightarrow$

$\quad\quad \langle \texttt{x}, \{x \mapsto 3, y \mapsto 5\}\rangle \Rightarrow 3$

$\quad\quad \langle \texttt{y}, \{x \mapsto 3, y \mapsto 5\}\rangle \Rightarrow 5$

$\quad\quad 3 > 5$ es false

$\quad$ *false*

$\quad \langle \texttt{m := y}, \{x \mapsto 3, y \mapsto 5\}\rangle \rightarrow$

$\quad\quad \langle \texttt{y}, \{x \mapsto 3, y \mapsto 5\}\rangle \Rightarrow 5$

$\quad \Downarrow \{x \mapsto 3, y \mapsto 5, m \mapsto 5\}$

$\Downarrow \{x \mapsto 3, y \mapsto 5, m \mapsto 5\}$

18. Calcula la precondición más débil $pmd(S, Q)$ para el programa S de la pregunta 15 y la postcondición Q dada por $\mathtt{x} = \mathtt{X} \wedge \mathtt{y} = \mathtt{Y}$.

$pmd((\mathtt{t} := \mathtt{x}; \mathtt{x} := \mathtt{y}; \mathtt{y} := \mathtt{t}), \mathtt{x} = \mathtt{X} \wedge \mathtt{y} = \mathtt{Y}) =$
$pmd(\mathtt{t} := \mathtt{x}, pmd(\mathtt{x} := \mathtt{y}, pmd(\mathtt{y} := \mathtt{t}, \mathtt{x} = \mathtt{X} \wedge \mathtt{y} = \mathtt{Y}))) =$ [step 1]
$pmd(\mathtt{t} := \mathtt{x}, pmd(\mathtt{x} := \mathtt{y}, \mathtt{x} = \mathtt{X} \wedge \mathtt{t} = \mathtt{Y})) =$ [step 2]
$pmd(\mathtt{t} := \mathtt{x}, \mathtt{y} = \mathtt{X} \wedge \mathtt{t} = \mathtt{Y}) =$ [step 3]
$\mathtt{y} = \mathtt{X} \wedge \mathtt{x} = \mathtt{Y}$

Para hacer el paso [step 1] se ha aplicado la regla para la *asignación* para calcular la pmd más interna, en particular

$$pmd(\mathtt{y} := \mathtt{t}, \mathtt{x} = \mathtt{X} \wedge \mathtt{y} = \mathtt{Y}) = (\mathtt{x} = \mathtt{X} \wedge \mathtt{y} = \mathtt{Y})[\mathtt{y} \mapsto \mathtt{t}] = (\mathtt{x} = \mathtt{X} \wedge \mathtt{t} = \mathtt{Y})$$

Para hacer el paso [step 2] procedemos de forma similar para calcular la *pmd* más interna, en particular

$$pmd(\mathtt{x} := \mathtt{y}, \mathtt{x} = \mathtt{X} \wedge \mathtt{t} = \mathtt{Y}) = (\mathtt{x} = \mathtt{X} \wedge \mathtt{t} = \mathtt{Y})[\mathtt{x} \mapsto \mathtt{y}] = (\mathtt{y} = \mathtt{X} \wedge \mathtt{t} = \mathtt{Y})$$

Para hacer el paso [step 3] se ha aplicado otra vez la regla para la asignación, en particular

$$pmd(\mathtt{t} := \mathtt{x}, \mathtt{y} = \mathtt{X} \wedge \mathtt{t} = \mathtt{Y}) = (\mathtt{y} = \mathtt{X} \wedge \mathtt{t} = \mathtt{Y})[\mathtt{t} \mapsto \mathtt{x}] = (\mathtt{y} = \mathtt{X} \wedge \mathtt{x} = \mathtt{Y})$$

De acuerdo a los resultados obtenidos:

(a) ¿Qué hace este programa? ¿Cuál es la diferencia (si la hay) entre el programa S y el siguiente programa S' que usa la asignación múltiple introducida en la pregunta 14?

```
x, y := y, x
```

El programa S' intercambia los valores de las variables `x` e `y`.
Existen algunas diferencias entre S y S':

- Si usamos la semántica operacional, la ejecución del programa original hace que se alcance un estado en el que aparecen tres variables: `x`, `y` y la variable auxiliar `t`. Sin embargo, si ejecutamos la instrucción de asignación múltiple, alcanzaremos un estado en el que sólo apareceran las dos variables `x` e `y`. Por lo tanto, si usamos la semántica operacional podemos decir que los programas son diferentes.
- Además, operacionalmente, si desarrolláramos la traza de la semántica *small-step* de los dos programas, veríamos que la traza del programa original consta de varios pasos mientras que la traza para el programa con la asignación múltiple consta únicamente de un paso.

(b) Considerando la semántica axiomática, ¿hay alguna diferencia entre S y S' con respecto a la postcondición Q?

Si definiéramos la regla de la semántica axiomática para la asignación múltiple (de forma similar a la asignación simple) y calculáramos la *pmd* del programa que usa la asignación múltiple, obtendríamos el mismo resultado que para el programa original, es decir, $pmd(\mathtt{x,y\ :=\ y,x}, Q) = (\mathtt{y{=}X} \wedge \mathtt{x{=}Y})$. Esto significa que desde el punto de vista de la semántica axiomática, estos dos programas son equivalentes.

(c) ¿Son equivalentes los programas S y S' desde el punto de vista operacional o se podrían distinguir usando la semántica?

> Tal y como se ha visto en el primer apartado de esta pregunta, los programas son distinguibles desde el punto de vista de la semántica operacional de paso pequño.

19. Dado el siguiente programa S:

```
X:=X-1
```

y dada la precondición $P = (\texttt{X=1})$ y la postcondición $Q = (\texttt{X} \geq \texttt{0})$, ¿es correcto S con respecto a P y Q? Usa la semántica axiomática (precondición más débil) para la demostración y muestra los cálculos realizados para comprobar la corrección o no corrección.

> Primero computamos $pmd(S, Q)$:
>
> $pmd(\texttt{X:=X-1}, \texttt{X} \geq \texttt{0}) = (\texttt{X} - \texttt{1} \geq \texttt{0}) \Leftrightarrow (\texttt{X} \geq \texttt{1})$
>
> Ahora, como $\texttt{X} = \texttt{1} \Rightarrow \texttt{X} \geq \texttt{1}$ claramente se cumple, la corrección de S con respecto a P y Q queda probada.

20. Calcula la precondición más débil de los siguientes programas teniendo en cuenta la postcondición indicada en cada caso:

(a) $S = (\texttt{x:=1})$, $Q = (\texttt{x} = \texttt{1})$.

> $pmd(\texttt{x:=1}, \texttt{x=1}) = (\texttt{1=1}) = \mathit{true}$

(b) $S = (\texttt{x:=y})$, $Q = (\texttt{x} = \texttt{0})$.

> $pmd(\texttt{x:=y}, \texttt{x=0}) = (\texttt{y=0})$

(c) $S = (\texttt{x:=x-1})$, $Q = (\texttt{x} = \texttt{0})$.

> $pmd(\texttt{x:=x-1}, \texttt{x=0}) = (\texttt{x-1=0}) = (\texttt{x=1})$

(d) $S = (\texttt{x:=x-1})$, $Q = (\texttt{y} > \texttt{0})$.

> $pmd(\texttt{x:=x-1}, \texttt{y>0}) = (\texttt{y>0})$

(e) $S = (\texttt{if (x>0) then x:=y else y:=x})$, $Q = (\texttt{x} \geq \texttt{y} \wedge \texttt{y} > \texttt{0})$.

> $pmd(\texttt{if } (\texttt{x} > \texttt{0}) \texttt{ then } \texttt{x} := \texttt{y} \texttt{ else } \texttt{y} := \texttt{x}, (\texttt{x} \geq \texttt{y} \wedge \texttt{y} > \texttt{0})) =$
> $(\texttt{x} > \texttt{0} \wedge pmd(\texttt{x} := \texttt{y}, (\texttt{x} \geq \texttt{y} \wedge \texttt{y} > \texttt{0}))) \vee (\texttt{x} \leq \texttt{0} \wedge pmd(\texttt{y} := \texttt{x}, (\texttt{x} \geq \texttt{y} \wedge \texttt{y} > \texttt{0}))) =$
> $(\texttt{x} > \texttt{0} \wedge \texttt{y} \geq \texttt{y} \wedge \texttt{y} > \texttt{0}) \vee (\texttt{x} \leq \texttt{0} \wedge \texttt{x} \geq \texttt{x} \wedge \texttt{x} > \texttt{0}) =$
> $(\texttt{x} > \texttt{0} \wedge \texttt{y} > \texttt{0}) \vee (\texttt{x} \leq \texttt{0} \wedge \texttt{x} > \texttt{0}) =$
> $(\texttt{x} > \texttt{0} \wedge \texttt{y} > \texttt{0})$

(f) $S = (\texttt{if (x=0) then x:=1 else x:=x+1})$, $Q = (\texttt{x} > \texttt{y} \wedge \texttt{y} \leq \texttt{0})$.

> $pmd(\texttt{if } (\texttt{x} = \texttt{0}) \texttt{ then } \texttt{x} := \texttt{1} \texttt{ else } \texttt{x} := \texttt{x} + \texttt{1}, (\texttt{x} > \texttt{y} \wedge \texttt{y} \leq \texttt{0})) =$
> $(\texttt{x} = \texttt{0} \wedge pmd(\texttt{x} := \texttt{1}, (\texttt{x} > \texttt{y} \wedge \texttt{y} \leq \texttt{0}))) \vee (\texttt{x} \neq \texttt{0} \wedge pmd(\texttt{x} := \texttt{x} + \texttt{1}, (\texttt{x} > \texttt{y} \wedge \texttt{y} \leq \texttt{0}))) =$
> $(\texttt{x} = \texttt{0} \wedge \texttt{1} > \texttt{y} \wedge \texttt{y} \leq \texttt{0}) \vee (\texttt{x} \neq \texttt{0} \wedge \texttt{x} + \texttt{1} > \texttt{y} \wedge \texttt{y} \leq \texttt{0}) =$
> $(\texttt{x} = \texttt{0} \wedge \texttt{y} \leq \texttt{0}) \vee (\texttt{x} \neq \texttt{0} \wedge \texttt{x} > \texttt{y} - \texttt{1} \wedge \texttt{y} \leq \texttt{0}) =$
> $\texttt{y} \leq \texttt{0} \wedge (\texttt{x} = \texttt{0} \vee (\texttt{x} \neq \texttt{0} \wedge \texttt{x} > \texttt{y} - \texttt{1})) =$
> $\texttt{y} \leq \texttt{0} \wedge (\texttt{x} = \texttt{0} \vee (\texttt{x} \neq \texttt{0} \wedge \texttt{x} \geq \texttt{y}))$

(g) $S = (\texttt{x:=y; y:=5})$, $Q = (\mathtt{x} > 0)$.

$$pmd(\mathtt{x} := \mathtt{y}; \mathtt{y} := 5, \mathtt{x} > 0) =$$
$$pmd(\mathtt{x} := \mathtt{y}, pmd(\mathtt{y} := 5, \mathtt{x} > 0)) =$$
$$pmd(\mathtt{x} := \mathtt{y}, \mathtt{x} > 0) = (\mathtt{y} > 0)$$

(h) $S = (\texttt{x:=x+1; if (x>0) then x:=y else y:=x})$, $Q = (\mathtt{x} > \mathtt{z})$.

$$pmd((\mathtt{x} := \mathtt{x} + 1; \mathtt{if}\ (\mathtt{x} > 0)\ \mathtt{then}\ \mathtt{x} := \mathtt{y}\ \mathtt{else}\ \mathtt{y} := \mathtt{x}), \mathtt{x} > \mathtt{z}) =$$
$$pmd(\mathtt{x} := \mathtt{x} + 1, pmd(\mathtt{if}\ (\mathtt{x} > 0)\ \mathtt{then}\ \mathtt{x} := \mathtt{y}\ \mathtt{else}\ \mathtt{y} := \mathtt{x}, \mathtt{x} > \mathtt{z})) =$$
$$pmd(\mathtt{x} := \mathtt{x} + 1, ((\mathtt{x} > 0 \wedge pmd(\mathtt{x} := \mathtt{y}, \mathtt{x} > \mathtt{z})) \ \vee\ (\mathtt{x} \leq 0 \wedge pmd(\mathtt{y} := \mathtt{x}, \mathtt{x} > \mathtt{z}))) =$$
$$pmd(\mathtt{x} := \mathtt{x} + 1, ((\mathtt{x} > 0 \ \wedge\ \mathtt{y} > \mathtt{z}) \ \vee\ (\mathtt{x} \leq 0 \ \wedge\ \mathtt{x} > \mathtt{z})) =$$
$$(\mathtt{x} + 1 > 0 \ \wedge\ \mathtt{y} > \mathtt{z}) \ \vee\ (\mathtt{x} + 1 \leq 0 \ \wedge\ \mathtt{x} + 1 > \mathtt{z}) =$$
$$(\mathtt{x} \geq 0 \ \wedge\ \mathtt{y} > \mathtt{z}) \ \vee\ (\mathtt{x} < 0 \ \wedge\ \mathtt{x} \geq \mathtt{z})$$

Parte II: Test

21. $\boxed{\text{B}}$
22. $\boxed{\text{A}}$
23. $\boxed{\text{B}}$
24. $\boxed{\text{A}}$
25. $\boxed{\text{C}}$
26. $\boxed{\text{A}}$
27. $\boxed{\text{D}}$
28. $\boxed{\text{C}}$
29. $\boxed{\text{A}}$
30. $\boxed{\text{A}}$
31. $\boxed{\text{B}}$
32. $\boxed{\text{C}}$
33. $\boxed{\text{D}}$
34. $\boxed{\text{B}}$
35. $\boxed{\text{D}}$
36. $\boxed{\text{A}}$

A Gramática BNF para C-Minus

```
<ID>                  ::= <letter><letter>*
<NUM>                 ::= <digit><digit>*
<letter>              ::= a | ...| z | A | ...| Z
<digit>               ::= 0 | ...| 9
<program>             ::= <declaration-list>
<declaration-list>    ::= <declaration-list> <declaration> | <declaration>
<declaration>         ::= <var-declaration> | <fun-declaration>
<var-declaration>     ::= <type-specifier> <ID> ; | <type-specifier> <ID> [ <NUM> ] ;
<type-specifier>      ::= int | void
<fun-declaration>     ::= <type-specifier> <ID> ( <params> ) <compound-stmt>
<params>              ::= <param-list> | void
<param-list>          ::= <param-list> , <param> | <param>
<param>               ::= <type-specifier> <ID> | <type-specifier> <ID> [ ]
<compound-stmt>       ::= { <local-declarations> <statement-list> }
<local-declarations> ::= <local-declarations> <var-declaration> | empty
<statement-list>      ::= <statement-list> <statement> | empty
<statement>           ::= <expression-stmt> | <compound-stmt> | <selection-stmt>
                          | <iteration-stmt> | <return-stmt>
<expression-stmt>     ::= <expression> ; | ;
<selection-stmt>      ::= if ( <expression> ) <statement>
                          | if ( <expression> ) <statement> else <statement>
<iteration-stmt>      ::= while ( <expression> ) <statement>
<return-stmt>         ::= return ; | return <expression> ;
<expression>          ::= <var> = <expression> | <simple-expression>
<var>                 ::= <ID> | <ID> [ <expression> ]
<simple-expression>   ::= <additive-expression> <relop> <additive-expression>
                          | <additive-expression>
<relop>               ::= <= | < | > | >= | == | !=
<additive-expression>::= <additive-expression> <addop> <term> | <term>
<addop>               ::= + | -
<term>                ::= <term> <mulop> <factor> | <factor>
<mulop>               ::= * | /
<factor>              ::= ( <expression> ) | <var> | <call> | <NUM>
<call>                ::= <ID> ( <args> )
<args>                ::= <arg-list> | empty
<arg-list>            ::= <arg-list> , <expression> | <expression>
```