Examen Parcial de PRED

1. **(1.5 puntos)** Dado el siguiente esquema de programa:

```
acción P1 (X: entero)
    var X1: entero
    acción P2 (X: entero)
         var X2: entero
    …
    facción
    acción P4 (X: entero)
         var X4: entero
         acción P5 (X: entero)
              var X5: entero
              acción P3 (X: entero)
                   var X3: entero

              …
              facción

         …
         facción
    …
    facción
…
facción
```

¿Cuales de las siguientes llamadas podrían estar en la acción P5 y cuales no (indicando el por qué)?

a) P1(X4)
b) P2 (X5)
c) P4 (X2)
d) P3 (X3)

2. **(1.5 puntos)** Dadas las declaraciones:

```
    clase C
          x: entero;
          acción P(y: entero)
          ...
          facción
    fclase

    clase C1 subclase de C
          x1: entero;
          acción P1(y:entero)
          ...
          facción
```

```
fclase

clase C2 subclase de C1
       x2: entero;
       acción P(y: entero)
       ...
       facción
       acción P2(y:entero)
       ...
       facción
fclase

clase C3 subclase de C
       x2: entero;
       x3: entero;
       acción P(y: entero)
       ...
       facción
       acción P3(y:entero)
       ...
       facción
fclase
```

Supongamos que hemos declarado las variables a: C; a1: C1; a2: C2; a3: C3 y supongamos que v, v1, v2 y v3 son, respectivamente, valores de las clases C, C1, C2 y C3. En este contexto, cuales de las siguientes secuencias de instrucciones serían correctas y por qué:

a) a:=v2; a.P1(a.x)
b) a2:= v; a1:= v1; a2.P(a1.x)
c) a:= v3; a.P(a2.x2)
d) a1:=v1; a1.P(a1.x);
e) a1:=v2; a1.P(a1.x);
Además, en el caso de las llamadas que consideres correctas al método P, ¿a qué P se llamaría? ¿al de C al de C2 o al de C3?

3. **(1.5 puntos)** Dadas las siguientes definiciones de tipos:

```
f: real x nat --> nat
f: nat x real --> int
g: real x real --> int
g: nat x int  --> nat
g: nat x real --> nat
```

siendo `nat` subtipo de `int`, e `int` subtipo de `real`. Decir si podemos inferir lo siguiente (y cómo):

```
a) f(g(2,-2),g(2,2.5)): real
b) f(g(2.5,2),g(-2,2.5)): int
```

sabiendo que, obviamente, `2: nat, -2: int, 2.5: real`.

4. **(1 punto)** Supongamos que en Java tenemos las siguientes declaraciones:

```
class C {
        ...
}

class C1 extends C {
        ...
}
```

¿Ves algún problema en las siguientes declaraciones?

```
a) class C2 {
     void P(C x, C1 y) {...}
     void P(C1 x, C y) {...}
   }

b) class C2 {
     void P(C x, C y) {...}
     void P(C1 x, C1 y) {...}
   }
```

Dadas además las declaraciones:

```
C1 v1;
C2 v2;
```

y dada la llamada

```
v2.P(v1,v1);
```

¿a qué método se llamaría en el caso a)? ¿Y en el caso b)?

5. **(1.5 puntos)** Dado el siguiente programa

```
acción P1 (Y: entero)
    var A,X: entero
    acción P2 (X:entero)
        var Y: entero
        A:= X+1;
        Y:= A+1;
        X:= Y+X;
    facción
    A:= 3; X:= 0;
    P2(A);
    escribir (A); escribir(X); escribir(Y);
facción
```

Suponiendo que se ejecutan las instrucciones B:= 4; P1(B), ¿qué se escribirá al final de la ejecución de P1 si el paso de parámetros se produce por valor-resultado? ¿y si se produce por referencia?.

6. **(3 puntos)** Especificar los números binarios como secuencias de bits. En particular, se ha de especificar una abstracción de datos con las siguientes operaciones:

   crear: dado un bit (un 0 o un 1) nos devuelve el número binario formado por ese único bit.
   añadir: dado un número binario B y un bit b nos devuelve la secuencia formada (de izquierda a derecha) por todos los bits de B seguidos de b.
   sumar: dados dos números B1 y B2, nos devuelve la secuencia que representa su suma.
   valor: dado un número binario B1, nos devuelve el número natural que representa. Para esta operación podemos usar las operaciones usuales sobre los números naturales.

Más concretamente, se pide:

a) Decir qué posibles conjuntos de constructores podríamos elegir.
b) Escribir la especificación completa de la abstracción.

Soluciones

1)

a) P1(X4): Si, la acción P1 es visible desde P5 y la variable X4 también.
b) P2 (X5): Sí, tanto la acción P2 como la variable X5 son visibles desde P5
c) P4 (X2): No, la variable X2 es local a P2 e invisible desde P5.
d) P3 (X3): No, X3 no es visible desde P5

2)

a) a:=v2; a.P1(a.x): La asignación a:=v2 es correcta. El objeto a toma un valor de una subclase suya. La llamada al método a.P1(a.x) será correcta o incorrecta dependiendo del lenguaje. Si la asignación de tipos es estrictamente dinámica, despues de la asignación anterior a tendría tipo C2. Como consecuencia sería correcta la llamada. Si la asignación de tipos es estática, el tipo de a sería C y, como consecuencia, la llamada sería incorrecta.
b) a2:= v; a1:= v1; a2.P(a1.x): Incorrecta. En la primera asignació no se puede asignar a una variable un valor de un supertipo suyo.
c) a:= v3; a.P(a2.x2); Perfectamente correcta. Como el valor de a es de tipo C3, normalmente el P llamado sería el de C3.
d) a1:=v1; a1.P(a1.x); Perfectamente correcta. El método llamado sería el P de la clase C.
e) a1:=v2; a1.P(a1.x); Perfectamente correcta. Como el valor de a es de tipo C2, normalmente el P llamado sería el de C2.

3)

a) Por una parte:

$$\dfrac{2: \text{nat} \qquad \dfrac{-2:\text{int} \quad \text{int}<\text{real}}{-2:\text{real}} \qquad g: \text{nat} \times \text{real} \to \text{nat}}{g(2,-2):\text{nat}}$$

Por otra:

$$\dfrac{\dfrac{2:\text{nat} \quad 2.5:\text{real} \quad g: \text{nat} \times \text{real} \to \text{nat}}{g(2,2.5):\text{nat}} \qquad \text{nat}<\text{real}}{g(2,2.5):\text{real}}$$

Finalmente:

$$\dfrac{g(2,-2):\text{nat} \quad g(2,2.5):\text{real} \quad f: \text{nat} \times \text{real} \to \text{int}}{f(g(2,-2),g(2,2.5)): \text{int}} \qquad \text{int}<\text{real}$$

$$f(g(2,-2),g(2,2.5)): \text{real}$$

b) `f(g(2.5,2),g(-2,2.5)): int` no puede ser inferido. En concreto, la única declaración de g que tiene un real como primer parámetro es `g: real x real --> int.` Como consecuencia, podremos demostrar que `g(2.5,2):int` (y, por tanto, `g(2.5,2):real`). Por otra parte, la única declaración de f que tiene un entero o un real como primer parámetro es `f: real x nat --> nat.` Como consecuencia, para demostrar que `f(g(2.5,2),g(-2,2.5)): int` tendríamos que demostrar que `g(-2,2.5): nat`. Pero esto no puede ser ya que las declaraciones de g que devuelven un resultado de tipo nat exigen que su primer parámetro tenga tipo también nat.

4)

La declaración a) de C2 tiene problemas de ambigüedad. En concreto, la llamada `v2.P(v1,v1);` podría ser resuelta, en principio, tanto con la primera declaración de P como con la segunda (si tenemos declarado un parámetro de clase C, siempre podemos pasar un argumento de clase C1 que es subclase de C). Pero ninguna regla de Java nos diría cual de las dos se ha de usar, lo que generaría un error (por la ambigüedad) que sería detectado por el compilador.
En cambio, la declaración b) de C2 es perfectamente correcta. En concreto, aunque la llamada `v2.P(v1,v1);` podría ser resuelta con cualquiera de las dos definiciones de P, se consideraría que la que se ha de elegir es la segunda, ya que los tipos de los parámetros y de los argumentos se ajustan mejor que en el caso de la primera definición.

5)

Paso por valor-resultado: escribiría 8 0 4
Paso por referencia: escribiría 9 0 4

6)

a) Podríamos elegir dos conjuntos distintos de constructores. Por una parte, crear y añadir nos permiten definir todos los números binarios. Por otra, también podemos definirlos todos con crear y sumar.
b) Una especificación simple que cumple con lo que se pide en el enunciado es la siguiente:

```
especificación num_binario
tipos numb, nat
operaciones
crear: bit -> numb
añadir: numb x bit -> numb
sumar: numb x numb -> numb
valor: numb -> nat
axiomas
sumar(B,crear(0))= B
sumar(crear(0),crear(1))= crear(1)
sumar(crear(1),crear(1))= añadir(crear(1),0)
sumar(añadir(B,0),crear(1))= añadir(B,1)
sumar(añadir(B,1),crear(1))= añadir(sumar(B,1),0)
sumar(añadir(B1,b1),añadir(B2,b2))=
sumar(añadir(sumar(B1,B2),b1), b2)
valor(crear(0))=0
valor(crear(1))=1
valor(sumar(B1,B2))= valor(B1)+valor(B2)
fespecificación
```