Metodología de la Programación

EXAMEN FINAL
23 de Mayo de 2011

Facultad de Informática de San Sebastián

APELLIDOS: .....

NOMBRE: .....

1. (3 puntos) Documentar con las aserciones que se marcan el siguiente programa que calcula el valor que resulta de restar uno al resultado de elevar dos a un numero natural dado.

(1) Pre {

```
z:=1; k:=n; r:=0;
```

(2) Inv {

while `k ≠ 1` loop (3) { E ≡

(4){

```
r:=z+r;
```

(5){

```
z:=2*z;
```

(6){

```
k:=k-1;
```

end loop;

(7){

```
r:=z+r;
```

(8) Post {

2. (1 puntos) Escribir una fórmula de primer orden que exprese que un array $A(1..n)$ de enteros contiene la misma cantidad de elementos que son múltiplos de un x dado que de elementos que son divisores de un y dado.
   Por ejemplo, para $x = 2$ e $y = 15$, el array $(8, 7, 3, 5, 3)$ no cumple la propiedad de arriba, ya que hay un múltiplo de 2 (que es 8) y tres divisores de 15 (que son 3, 5 y 3), mientras que el array $(4, 3, 4, 7, 5)$ sí cumple la propiedad porque contiene dos múltiplos de 2 (que son 4 y 4) y dos divisores de 15 (que son 3 y 5).

3. (2 puntos) Demostrar formalmente que

```
{1<y≤n ∧ z*n!*(m-n)!=w*y!*(m-y)!}
   z := (z*(m-y+1));
   w := w*y;
   y:=y-1;
{1≤y≤n ∧ z*n!*(m-n)!=w*y!*(m-y)!}
```

4. (2 puntos) Especificar ecuacionalmente una función que dada una secuencia s, un elemento x y un número natural n, decida si x aparece en s al menos n veces. Por ejemplo, para $s = \langle a, b, d, a, e, d \rangle$ y $x = a$, si $n = 2$, el resultado sería $True$, mientras que si $n = 3$ sería $False$. Nótese que para $n = 1$ también daría $True$ y para $n = 4$ también $False$.

5. (2 puntos) Utilizando el método de Burstall, diseñar un algoritmo iterativo que compute la siguiente función `f: secuencia(T) × Positive × Positive → secuencia(T)` especificada ecuacionalmente por:

   (1) $\texttt{f}(\langle\,\rangle, \texttt{p}, \texttt{n}) = \langle\,\rangle$;

   (2) $\texttt{f}(\texttt{x} \bullet \texttt{s}, \texttt{p}, \texttt{n}) = \begin{cases} \texttt{x} \bullet \texttt{f}(\texttt{s}, \texttt{n}, \texttt{n}) & \texttt{si p} = 1 \\ \texttt{f}(\texttt{s}, \texttt{p-1}, \texttt{n}) @ \langle \texttt{x} \rangle & \texttt{e.o.c.} \end{cases}$

   o definida como función recursiva por:

   function `f (s:secuencia(T),p,n:Positive)` **return** `r:secuencia(T)` **is**
   if `es_vacia(s)` **then** $\texttt{s} := \langle\,\rangle$;
   elsif $\texttt{p} = 1$ **then** $\texttt{r} := \texttt{f}(\texttt{resto}(\texttt{s}), \texttt{n}, \texttt{n})$; $\texttt{r} := \texttt{prim}(\texttt{s}) \bullet \texttt{r}$;
   else $\texttt{r} := \texttt{f}(\texttt{resto}(\texttt{s}), \texttt{p-1}, \texttt{n})$; $\texttt{r} := \texttt{r} @ \langle \texttt{prim}(\texttt{s}) \rangle$;
   end if;

   INVARIANTE/RELACIÓN DE RECURRENCIA :

INICIALIZACIÓN :

FINALIZACIÓN :

CUERPO DE LA ITERACIÓN(DESPLEGADO/PLEGADO) :

FUNCIÓN DOCUMENTADA :