Parcial de Imperativo - Algoritmos y Estructuras de Datos I 30 nov. 2005

Aclaraciones: El parcial NO es a libro abierto. Cualquier decisión de interpretación que se tome debe ser aclarada y justificada. Para aprobar el parcial se requieren al menos 60 puntos. Indicar el número de orden, LU y la cantidad total de hojas entregadas. Entregar cada ejercicio en hoja separada.

Ejercicio 1. Dada la siguiente especificación:

problema modificaSegúnHayaUnInversoAntes($l : [\mathsf{Float}], n : \mathbb{Z}$) {
  modifica l;
  requiere $|l| == n$;
  asegura $|l| == |\mathsf{pre}(l)|$;
  asegura $l == [\mathsf{modif}(\mathsf{pre}(l), i) \mid i \leftarrow [0..|l|)]$;
  aux $\mathsf{modif}(y : [\mathsf{Float}], k : \mathbb{Z}) : \mathsf{Float} = \beta(\mathsf{hayInv}(y, k))2y_k + \beta(\neg\mathsf{hayInv}(y, k))y_k$;
  aux $\mathsf{hayInv}(y : [\mathsf{Float}], k : \mathbb{Z}) : \mathsf{Bool} = (\exists j \leftarrow [0..k))\ y_k == -y_j$;
}

1. **[20 p.]** Implemente la función en imperativo.
2. Para cada ciclo utilizado en la implementación del item anterior, proponga:
   a) **[5 p.]** una poscondición para el ciclo
   b) **[20 p.]** un invariante
   c) **[5 p.]** una expresión variante

   que permitan demostrar correctitud utilizando el Teorema del Invariante (¡no se pide demostrar nada!).

¡Sigue del otro lado! ↻

Ejercicio 2. Dada la siguiente especificación y su correspondiente implementación:

problema sumar($a : [\mathbb{Z}], b : [\mathbb{Z}], c : [\mathbb{Z}], n : \mathbb{Z}$) = {
    modifica c;
    requiere $|a| == n \land |b| == n \land |c| == n + 1$;
    requiere todosDígitos(a) $\land$ todosDígitos(b);
    asegura dec(c) == dec(a) + dec(b);
    asegura todosDígitos(c);
    aux dec($l : [\mathbb{Z}]$) : $\mathbb{Z} = \sum[10^{|l|-i-1} l_i \mid i \leftarrow [0..|l|)]$;
    aux todosDígitos($l : [\mathbb{Z}]$) : Bool = $(\forall x \leftarrow l)\ x \in [0..9]$;
}

```
int sumar (const int a[], const int b[], int c[], int n) {
    int mellevo = 0;
    int i = n - 1;
      // estado antesC;
      // vale i == n − 1 ∧ mellevo == 0;
    while ( i >= 0 ) {
          // invariante I : dec(a(i..n)) + dec(b(i..n)) == dec(c(i + 1..n + 1)) + 10^{n−i−1} mellevo ∧
          //                −1 ≤ i < n ∧ todosDígitos(c(i + 1..n + 1)) ∧ mellevo ∈ [0, 1]
          // variante (−1) v: i;
        if (a[i] + b[i] + mellevo < 10) {
            c[i + 1] = a[i] + b[i] + mellevo;
            mellevo = 0;
        } else {
            c[i + 1] = a[i] + b[i] + mellevo - 10;
            mellevo = 1;
        }
        i = i - 1;
    }
      // estado despuésC;
      // vale Q : dec(a) + dec(b) == dec(c(0..n + 1)) + 10^n mellevo ∧ todosDígitos(c(0..n + 1))∧
      //          mellevo ∈ [0, 1]
    c[0] = mellevo;
      // vale dec(a) + dec(b) == dec(c) ∧ todosDígitos(c)
}
```

Se desea completar la demostración de correctitud de esta implementación. Para ello se pide demostrar algunos puntos del Teorema del Invariante:

1. **[5 p.]** El invariante y la negación de la guarda garantizan que vale la poscondición del ciclo: $(I \land \neg B) \rightarrow Q$.
2. **[5 p.]** El invariante vale en *antesC*: $antesC \models I$.
3. **[5 p.]** La expresión variante está acotada: $(I \land v \leq -1) \rightarrow \neg B$.
4. **[35 p.]** El cuerpo del ciclo preserva el invariante: `/*` vale $I \land B$; `*/ cuerpo /*` vale I; `*/`.

¡Empieza del otro lado! ↺