Práctica 4
Especificación de problemas

Algoritmos y Estructura de Datos I

Primer Cuatrimestre 2011

Ejercicio 1. Las siguientes especificaciones no son correctas. Indicar por qué, y corregirlas para que describan correctamente el problema.

I) problema arcoSeno($x, \varepsilon : \mathbb{R}$) $= res : \mathbb{R}\{$
    requiere : $\varepsilon > 0$;
    asegura : $|\sin(res) - x| \leq \varepsilon$;
}

II) problema buscar($lista : [T], elem : T$) $= res : \mathbb{Z}\{$
    requiere : $elem \in lista$;
    asegura : $lista_{res} == elem$;
}

III) problema progresionGeometricaFactor2($l : [\mathbb{Z}]$) $= res :$ Bool$\{$
    asegura : $res == (\forall i \leftarrow [0..|l|))l_i == 2 \times l_{i-1}$;
}

IV) problema pivotear($lista : [\mathbb{Z}]$) $= res : [\mathbb{Z}]\{$
    asegura : $res == menores + +$cab$(lista) + +mayores$;
    aux $menores : [\mathbb{Z}] = [x|x \leftarrow lista, x <$ cab$(lista)]$;
    aux $mayores : [\mathbb{Z}] = [x|x \leftarrow lista, x >$ cab$(lista)]$;
}

V) problema raizEnesima($n : \mathbb{Z}, x, \varepsilon : \mathbb{R}$) $= res : \mathbb{R}\{$
    requiere : $\varepsilon > 0$;
    requiere : $n > 0$;
    asegura : $|res^n - x| \leq \varepsilon$;
}

VI) problema minimo($lista : [\mathbb{Z}]$) $= x : \mathbb{Z}\{$
    requiere : $|lista| > 0$;
    requiere : $(\forall x \leftarrow lista, y \leftarrow lista)x \neq y$;
    asegura : $x \in lista$;
    asegura : $(\forall y \leftarrow lista, y \neq x)y > x$;
}

Ejercicio 2. La siguiente no es una especificación válida, ya que para ciertos valores de entrada que cumplen la precondición, no existe una salida que cumpla con la postcondición.

problema subListaQueSume($lista : [\mathbb{Z}], suma : \mathbb{Z}) = sublista : [\mathbb{Z}]$\{
    asegura Incluida: $(\forall x \leftarrow sublista) cant(sublista, x) \leq cant(lista, x)$;
    asegura SumaJusta: $\sum sublista == suma$;
    aux $cant(l : [\mathbb{Z}], n : \mathbb{Z}) : \mathbb{Z} = |[x | x \leftarrow l, x == n]|$;
}

I) Mostrar valores para *lista* y *suma* que hagan verdadera la precondición, pero tales que no exista *sublista* que cumpla la postcondición.

II) Supongamos que agregamos a la especificación la siguiente cláusula:

    requiere : $min\_suma \leq suma \leq max\_suma$
    aux $min\_suma : \mathbb{Z} = \sum [x | x \leftarrow lista, x < 0]$
    aux $max\_suma : \mathbb{Z} = \sum [x | x \leftarrow lista, x > 0]$

    ¿Ahora es una especificación válida? Si no lo es, justificarlo con un ejemplo como en el punto anterior.

Ejercicio 3. Para los siguientes problemas, dar todas las soluciones posibles a las entradas dadas.

I) problema raizCuadrada($x : \mathbb{R}) = res : \mathbb{R}$\{
        requiere : $x \geq 0$;
        asegura : $res^2 == x$;
    }

    a) $x = 0$
    b) $x = 1$
    c) $x = 27$

II) problema indiceDelMaximo($l : [\mathbb{R}]) = res : \mathbb{Z}$\{
        requiere : $|l| > 0$;
        asegura : $0 \leq res < |l|$;
        asegura : $(\forall i \leftarrow [0..|l|)) l_i \leq l_{res}$;
    }

    a) $l = [1, 2, 3, 4]$
    b) $l = [15.5, -18, 4.215, 15.5, -1]$
    c) $l = [0, 0, 0, 0, 0, 0]$

III) problema indiceDelPrimerMaximo($l : [\mathbb{R}]) = res : \mathbb{Z}$\{
        requiere : $|l| > 0$;
        asegura : $0 \leq res < |l|$;
        asegura : $(\forall i \leftarrow [0..|l|)) (l_i < l_{res} \vee (l_i == l_{res} \wedge i \geq res))$;
    }

    a) $l = [1, 2, 3, 4]$
    b) $l = [15.5, -18, 4.215, 15.5, -1]$
    c) $l = [0, 0, 0, 0, 0, 0]$

¿Para qué valores de entrada indiceDelPrimerMaximo y indiceDelMaximo tienen necesariamente la misma salida?

Ejercicio 4. Sea $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ definida como:

$$f(a,b) = \begin{cases} 2b & \text{si } a < 0 \\ b-1 & \text{en otro caso} \end{cases}$$

¿Cuáles de las siguientes especificaciones son correctas para el problema de calcular $f(x)$? Para las que no lo son, indicar por qué.

I) problema f$(a, b : \mathbb{R}) = r : \mathbb{R}\{$
asegura : $a < 0 \land r == 2 \times b$;
asegura : $a \geq 0 \land r == b - 1$;
$\}$

II) problema f$(a, b : \mathbb{R}) = r : \mathbb{R}\{$
asegura : $(a < 0 \land r == 2 \times b) \lor (a > 0 \land r == b - 1)$;
$\}$

III) problema f$(a, b : \mathbb{R}) = r : \mathbb{R}\{$
asegura : $(a < 0 \land r == 2 \times b) \lor (a \geq 0 \land r == b - 1)$;
$\}$

IV) problema f$(a, b : \mathbb{R}) = r : \mathbb{R}\{$
asegura : $a < 0 \to r == 2 \times b$;
asegura : $a \geq 0 \to r == b - 1$;
$\}$

V) problema f$(a, b : \mathbb{R}) = r : \mathbb{R}\{$
asegura : $(a < 0 \to r == 2 \times b) \lor (a \geq 0 \to r == b - 1)$;
$\}$

VI) problema f$(a, b : \mathbb{R}) = r : \mathbb{R}\{$
asegura : $r ==$ (if $a < 0$ then $2 \times b$ else $b - 1$);
$\}$

Ejercicio 5. Considerar la siguiente especificación:

problema unoMasGrande$(x : \mathbb{R}) = res : \mathbb{R}\{$
asegura : $res > x$;
$\}$

y un algoritmo que dado x devuelve x^2.

I) ¿Qué devuelve el algoritmo si recibe $x = 3$? ¿El resultado hace verdadera la postcondición de unoMasGrande?

II) ¿Qué sucede para las entradas x = 0.5, x=1, x = -0.2 y x = -7?

III) Teniendo en cuenta lo respondido en los puntos anteriores, escribir una precondición para unoMasGrande, de manera tal que el algoritmo sea una implementación correcta.

Ejercicio 6. Considerar las siguientes dos especificaciones:

problema p1($x : \mathbb{R}, n : \mathbb{Z}$) = $res : \mathbb{Z}$\{
    requiere : $x \neq 0$;
    asegura : $x^n - 1 < res \leq x^n$;
}

problema p2($x : \mathbb{R}, n : \mathbb{Z}$) = $res : \mathbb{Z}$\{
    requiere : $n \leq 0 \rightarrow x \neq 0$;
    asegura : $res == \lfloor x^n \rfloor$;
}

Y un algoritmo a que satisface la especificación de p2.

I) Supongamos valores de x y n que hacen verdadera la precondición de p1. ¿Hacen también verdadera la precondición de $p2$?

II) Ahora, dados estos valores de x y n, supongamos que se ejecuta a: llegamos a un valor de res que hace verdadera la postcondición de p2. ¿Será también verdadera la postcondición de p1?

III) ¿Podemos concluir que a satisface la especificación de p1?

Ejercicio 7. Considerar las siguientes especificaciones:

problema n-esimo1($l : [T], n : Int$) = $res : \mathbb{Z}$\{
    requiere Ordenados: $(\forall i \leftarrow [0..|l| - 1))l_i < l_{i+1}$;
    requiere : $0 \leq n < |l|$;
    asegura : $res == l_n$;
}

problema n-esimo2($l : [T], n : Int$) = $res : \mathbb{Z}$\{
    requiere Distintos: $(\forall i \leftarrow [0..|l|), j \leftarrow [0..|l|), i \neq j)l_i \neq l_j$;
    requiere : $0 \leq n < |l|$;
    asegura : $n == |[x|x \leftarrow l, x < res]|$;
}

¿Es cierto que todo algoritmo que cumple con n-esimo1 cumple también con n-esimo2? ¿Y al revés?
Sugerencia: Razonar de manera análoga a la del ejercicio anterior.

Ejercicio 8. Especificar los siguientes problemas:

I) Dado un número entero, decidir si es par.

II) Dado un entero n y uno m, decidir si n es un múltiplo de m.

III) Dado un número real, devolver su inverso multiplicativo.

IV) Dada una lista de caracteres, obtener de ella sólo los que son numéricos (con todas sus apariciones).

V) Dada una lista de reales, obtener otra que resulte de aumentar en 1 los números de sus posiciones pares.

VI) Dado un número entero, listar todos sus divisores positivos (sin duplicados).

Ejercicio 9. Considerar el problema de decidir, dados n y m enteros, si n es múltiplo de m, y la siguiente especificación.

problema esMultiplo$(n, m : \mathbb{Z}) = res$: Bool{
    requiere : $m \neq 0$;
    asegura : $res == (n \mod m == 0)$;
}

I) Según la definición de múltiplo, ¿tiene sentido preguntarse si 4 es múltiplo de 0? ¿Cuál es la respuesta?

II) ¿Debería ser $n = 4$, $m = 0$ una entrada válida para el problema? ¿Lo es en esta especificación?

III) Corregir la especificación de manera tal que $n = 4$, $m = 0$ satisfaga la precondición (¡cuidado con las indefiniciones!).

IV) ¿Qué relación de fuerza hay entre la precondición nueva y la original?

Ejercicio 10. Considerar el problema de, dada una lista de números reales, devolver la que resulta de duplicar sus valores en las posiciones impares.

I) Para la lista $[1, 2, 3, 4]$, ¿es $[0, 4, 0, 8]$ un resultado correcto?

II) Sea la siguiente especificación:

problema duplicarEnImpares$(l : [\mathbb{R}]) = res : [\mathbb{R}]$\{
    asegura : $|res| == |l|$;
    asegura : $(\forall i \leftarrow [0..|res|), i \mod 2 == 1) res_i == 2 \times l_i$;
}

Si $l = [1, 2, 3, 4]$, ¿$res = [0, 4, 0, 8]$ satisface la postcondición?

III) Si es necesario, corregir la especificación para que describa correctamente el resultado esperado.

IV) ¿Qué relación de fuerza hay entre la nueva postcondición y la original?

Ejercicio 11. Para el problema de obtener, dado un entero positivo, la lista de todos sus factores primos (sin duplicados):

I) Dado el número 660 como entrada, ¿la lista $[3, 2, 5, 11]$ es un resultado válido?

II) Considerar la siguiente especificación:

problema factoresPrimos$(n : \mathbb{Z}) = fs : [\mathbb{Z}]$\{
    requiere : $n > 0$;
    asegura : $fs == [f | f \leftarrow [1..n], primo(f), n \mod f == 0]$;
    aux $primo(f : \mathbb{Z})$: Bool $= f > 1 \wedge \neg(\exists d \leftarrow [2..f)) f \mod d == 0$;
}

¿La lista $fs = [3, 2, 5, 11]$ satisface la postcondición, para $n = 660$?

III) Teniendo en cuenta lo respondido en los puntos anteriores, ¿puede decirse que la especificación del punto anterior describe correctamente el problema? Si no, corregir la postcondición de manera tal que sí lo haga.

IV) ¿Qué relación de fuerza hay entre la nueva postcondición y la propuesta en el punto II?

Ejercicio 12. Sea la siguiente especificación, para el problema de obtener de una lista de caracteres, sólo los que son una letra en minúsculas (con todas sus apariciones):

problema soloLetrasMinusculas(l : [Char]) = res : [Char]{
    asegura : $(\forall a \leftarrow l) a \in res \leftrightarrow a \in ['a'..'z']$;
}

I) Para $l = ['A','b','c','5']$, hay varios valores de res que hacen verdadera la postcondición, aunque no constituyen una solución correcta para el problema. Dar dos ejemplos.

II) Dar todos los valores de res correctos, para el mismo valor de l.

III) Escribir una postcondición correcta para este problema. ¿Qué relación de fuerza tiene con la original?

Ejercicio 13. Con lo visto en los ejercicios 9 a 12, ¿Encuentra casos de sub y sobreespecificación en las especificaciones del ejercicio 8?

Ejercicio 14. Especificar los siguientes problemas:

I) Dado un número entero positivo, obtener la suma de sus factores primos.

II) Dado un número entero positivo, decidir si es perfecto. Se dice que un número es perfecto cuando es igual a la suma de sus divisores (excluyéndose a sí mismo).

III) Dado un número entero positivo n, obtener el menor entero positivo $m > 1$ tal que m sea coprimo con n.

IV) Dado un entero positivo, obtener su descomposición en factores primos. Devolver una lista de tuplas (p, e), donde p es un factor primo y e es su exponente, ordenada en forma creciente con respecto a p.

V) Dada una lista de números reales, obtener la diferencia máxima entre dos de sus elementos.

VI) Dada una lista de números enteros, devolver aquel que divida a más elementos de dicha lista. El elemento tiene que pertenecer a la lista original.

Ejercicio 15. Especificar los siguientes problemas sobre listas:

I) problema nEsimaAparicion($l : [T], e : T, n : \mathbb{Z}$) = $res : Int$, que devuelve el índice de la n-ésima aparición de e en l, comenzando desde 1.

II) Dadas dos listas s y t, decidir si s es una subcadena de t.

III) Dadas dos listas s y t, decidir si s está *incluida* en t, es decir, si todos los elementos de s aparecen en t en igual o mayor cantidad.

IV) problema mezclarOrdenado($s, t : [\mathbb{Z}]$) = $res : [\mathbb{Z}]$, que recibe dos listas ordenadas y devuelve el resultado de intercalar sus elementos ordenadamente.

v) Dadas dos listas s y t, devolver su *intersección*, es decir, una lista con todos los elementos que aparecen en ambas, cada uno la máxima cantidad de veces que aparezca tanto en s como en t.

vi) Dadas dos listas s y t, devolver su *intersección ordenada*, es decir, una lista con todos los elementos que aparecen en ambas, cada uno la máxima cantidad de veces que aparezca tanto en s como en t, y en el orden en que aparecen en s.

Ejercicio 16. Especificar los siguientes problemas:

i) problema cantApariciones($l : [T]$) = $res : [(T, \mathbb{Z})]$, que devuelve la lista con todos los elementos de l, sin duplicados y en el mismo orden, cada uno junto con su cantidad de apariciones.

ii) Dada una lista, devolver otra con todos sus prefijos, en orden creciente de longitud.

iii) problema concatMultiple($ls : [[T]]$) = $l : [T]$, que devuelve en l el resultado de concatenar todas las listas de ls, en el orden en que aparecen.

iv) Dada una lista de listas de enteros l, devolver una lista de l que contenga el máximo valor. Por ejemplo, si l = [[2,3,5] , [8,1] , [2,8,4,3]], devolver [8,1] o [2,8,4,3].

v) problema interseccionMultiple($ls : [[T]]$) = $l : [T]$, que devuelve en l el resultado de la intersección de todas las listas de ls. Es importante no imponer un orden.

vi) problema separar($l : [T], delim : T$) = $ls : [[T]]$, que devuelve la lista resultante de separar l en cada posición donde aparece el delimitador $delim$ (y eliminar dicho delimitador). Por ejemplo, separar($['h','o','l',a',';','a','m','i','g','o',';','v','e','n']$,';') = $[['h','o','l',a'], ['a','m','i','g','o'], ['v','e','n']]$.

vii) Dada una lista l con todos sus elementos distintos, devolver la lista de *partes*, es decir, la lista de todas las listas incluidas en l, cada una con sus elementos en el mismo orden en que aparecen en l.

Especificación de problemas usando **modifica**

Ejercicio 17. Dados dos enteros a y b, se necesita calcular su suma, y dejarla almacenada en un entero c. ¿Cuáles de las siguientes especificaciones son correctas para este problema? Para las que no lo son, indicar por qué.

i)
```
problema sumar(a, b, c : ℤ){
    modifica a + b;
    asegura : a + b == c;
}
```

ii)
```
problema sumar(a, b, c : ℤ){
    asegura : c == a + b;
}
```

iii)
```
problema sumar(a, b, c : ℤ){
    modifica c;
    asegura : c == a + b;
}
```

IV) problema sumar($a, b, c : \mathbb{Z}$){
    modifica c;
    asegura : $a ==$ pre(a);
    asegura : $c == a + b$;
}

Ejercicio 18. Dada una lista l, se desea sacar su primer elemento y devolverlo. Decidir cuáles de estas especificaciones son correctas. Para las que no lo son, indicar por qué y justificar con ejemplos.

I) problema tomarPrimero($l : [T]$) $= res : T$ {
    requiere NoVacia: $|l| > 0$;
    modifica l;
    asegura : $res ==$ cab(l);
}

II) problema tomarPrimero($l : [T]$) $= res : T$ {
    requiere NoVacia: $|l| > 0$;
    modifica l;
    asegura : $res ==$ cab(pre(l));
}

III) problema tomarPrimero($l : [T]$) $= res : T$ {
    requiere NoVacia: $|l| > 0$;
    modifica l_0;
    asegura : $res ==$ cab(pre(l));
    asegura : $|l| == |$pre$(l)| - 1$;
}

IV) problema tomarPrimero($l : [T]$) $= res : T$ {
    requiere NoVacia: $|l| > 0$;
    modifica l;
    asegura : $res ==$ cab(pre(l));
    asegura : $l ==$ cola(pre(l));
}

V) problema tomarPrimero($l : [T]$) $= res : T$ {
    requiere NoVacia: $|l| > 0$;
    modifica l;
    asegura : $res ==$ cab(pre(l));
    asegura : $l ==$ pre$(l)[1..|l|)$;
}

Ejercicio 19. Considerar la siguiente especificación:

problema intercambiar($l : [T], i, j : \mathbb{Z}$){
  requiere : $0 \leq i < |l|$;
  requiere : $0 \leq j < |l|$;
  modifica l;
  asegura MismaLong: $|l| == |\text{pre}(l)|$;
  asegura Intercambio_i: $l_i == \text{pre}(l_j)$;
  asegura Intercambio_j: $l_j == \text{pre}(l_i)$;
}

I) ¿Esta especificación es válida? Si lo es, ¿qué problema describe?

II) Mostrar con un ejemplo que la postcondición está sub-especificada (es decir, que hay valores que la hacen verdadera aunque no son deseables como solución).

III) Corregir la especificación agregando a la postcondición una o más cláusulas asegura : .

Ejercicio 20. Explicar coloquialmente la siguiente especificación:

problema copiarPrimero($l : [\mathbb{Z}], i : \mathbb{Z}$){
  requiere NoVacia: $|l| > 0$;
  requiere EnRango: $0 \leq i < |l|$;
  modifica i;
  modifica l;
  asegura : $l_{\text{pre}(i)} = \text{pre}(l)_0$;
  asegura : $i = \text{pre}(l)_{\text{pre}(i)}$;
  asegura : $(\forall j \leftarrow [0..|l|), j \neq \text{pre}(i))l_j == \text{pre}(l)_{\text{pre}(i)}$;
}

Ejercicio 21. Dada una lista de enteros, se requiere duplicar aquéllos que se encuentran en posiciones pares. Indicar por qué son incorrectas las siguientes especificaciones, y proponer una alternativa correcta.

I) problema duplicarPares($l : [\mathbb{Z}]$){
    modifica l;
    asegura MismaLong: $|l| == |\text{pre}(l)|$;
    asegura : $(\forall i \leftarrow [0..|l|), i \mod 2 == 0)l_i == 2 \times \text{pre}(l_i)$;
  }

II) problema duplicarPares($l : [\mathbb{Z}]$){
    modifica $[l_i | i \leftarrow [0..|l|), i \mod 2 == 0]$;
    asegura : $(\forall i \leftarrow [0..|l|), i \mod 2 == 0)l_i == 2 \times \text{pre}(l_i)$;
  }

III) problema duplicarPares($l : [\mathbb{Z}]$){
    $(\forall i \leftarrow [0..|l|), i \mod 2 == 0)$ modifica l_i
    asegura : $(\forall i \leftarrow [0..|l|), i \mod 2 == 0)l_i == 2 \times \text{pre}(l_i)$;
  }

Ejercicio 22. Especificar los siguientes problemas de modificación de listas:

I) problema primosHermanos($l : [\mathbb{Z}]$), que dada una lista de enteros mayores a dos, reemplaza dichos valores por el número primo menor más cercano. Por ejemplo, si $l = [6, 5, 9, 14]$, luego de aplicar primosHermanos(l), $l = [5, 5, 7, 13]$

II) problema reemplazar($l : [T], a, b : T$), que reemplaza todas las apariciones de a en l por b.

III) problema recortar($l : [T], a : T$), que saca de l todas las apariciones de a consecutivas que aparezcan al principio. Por ejemplo recortar($[2, 2, 3, 2, 4], 2$) $= [3, 2, 4]$, mientras que recortar($[2, 2, 3, 2, 4], 3$) $= [2, 2, 3, 2, 4]$.

IV) problema intercambiarParesConImpares($l : [T]$), que toma una lista de longitud par y la modifica de modo tal que todas las posiciones de la forma $2k$ quedan intercambiadas con las posiciones $2k+1$. Por ejemplo, intercambiarParesConImpares($['a', 'd', 'i', 'n', 'l', 'e'] = ['d', 'a', 'n', 'i', 'e', 'l']$)

V) problema limpiarDuplicados($l : [T]$) $= dup : [T]$, que elimina los elementos duplicados de l dejando sólo su primera aparición, y devuelve en dup todas las apariciones eliminadas (respetando el orden).

Ejercicios tipo parcial

Ejercicio 23. Especificar los siguientes problemas

I) problema escaleraMasLarga($l : [\mathbb{Z}]$) $= escalera : (\mathbb{Z}, \mathbb{Z})$, que devuelve las posiciones de inicio y fin de la subcadena de l que constituya la escalera más larga. Una escalera es una secuencia de números donde cada uno es el sucesor del anterior. Por ejemplo:

- escaleraMasLarga($[-1, 0, 1, 2, 3, 5, 7, 8, 9, 0, 4]$) es $(0, 4)$.
- escaleraMasLarga($[-1, 0, 1, 5, 7, 8, 9, 0, 4]$) puede ser $(0, 2)$ o $(4, 6)$.
- escaleraMasLarga($[1, 3, 4, 9, 1]$) es $[(1, 2)]$.

II) problema picos($l : [\mathbb{Z}]$) $= res : [\mathbb{Z}]$, que devuelve la lista con las posiciones de los *picos* de l, en orden ascendente. Un *pico* es un elemento de la lista que es mayor estricto que el elemento que está a su derecha (si es que tiene) y a su izquierda (si es que tiene). Las posiciones de l se numeran desde 0. Por ejemplo:

- picos($[\mathbf{1}, 0, \mathbf{9}, 3, 7, \mathbf{8}, 5, 3, 3, 3, 2, \mathbf{7}, 1]$) $= [0, 2, 5, 11]$
- picos($[\mathbf{6}, 4, 2, 2, \mathbf{3}]$) $= [0, 4]$

III) problema picosNoCrecientes($L : [\mathbb{Z}]$) $= res : [\mathbb{Z}]$, que devuelve la lista con las posiciones de los picos de l que sean más bajos que algún pico anterior. Por ejemplo:

- picosNoCrecientes($[\not{\mathbf{1}}, 0, \not{\mathbf{9}}, 3, 7, \mathbf{8}, 5, 3, 3, 3, 2, \mathbf{7}, 1]$) $= [5, 11]$ porque el 8 es más bajo que el 9 y el 7 es más bajo que el 9, por ejemplo. Sin embargo, el 9 no es más bajo que el 1 y el 1 no tiene ningún pico anterior.
- picosNoCrecientes($[\not{\mathbf{6}}, 4, 2, 2, \not{\mathbf{3}}]$) $= [4]$

Ejercicio 24. Especificar los siguientes problemas:

I) Dada una lista de enteros, ordenarla de forma tal que los números pares antecedan a los impares, los pares ordenados en forma creciente y los impares en forma decreciente. Por ejemplo, al ordenar la lista $[1, 5, 8, 7, 9, -1, -4, 0]$ debería quedar $[-4, 0, 8, 9, 7, 5, 1, -1]$.

II) problema partirAlMedio($l : [\mathbb{R}], m : \mathbb{Z}, r :$ Bool), que decida si es posible partir l en dos listas (sin alterar el orden) de forma tal que sus elementos sumen lo mismo. En caso afirmativo, r debe ser verdadero y m debe contener la longitud de la primera de las listas. En caso contrario, r debe ser falso y m debe permanecer sin cambios. Por ejemplo:

- $[4, 5, 4, 6, 7]$ se puede dividir en $[4, 5, 4]$ y $[6, 7]$, donde ambas partes suman 13. En este caso, r deberá ser verdadero y m deberá quedar con el valor 3.
- $[10, 2, 1, 2, 5]$ se puede dividir en $[10]$ y $[2, 1, 2, 5]$, donde ambas partes suman 10. Aquí, r deberá ser verdadero y m deberá ser 1.
- $[-1{,}5, 4, -2{,}5]$ se puede dividir en $[\,]$ y $[-1{,}5, 4, -2{,}5]$ (o al revés), donde ambas partes suman 0. En ese caso, r deberá ser verdadero y m deberá valer 0 o 3.
- $[1, 3, 6]$ no puede ser dividido en dos de manera tal que cada parte sume lo mismo. En este caso, r deberá ser falso y m mantendrá su valor.

III) problema esSubLista$(l, s : [T], is : [\mathbb{Z}]) = r :$ Bool, que decide si s es una sub-lista incluida en l. En caso afirmativo, r será verdadero y en is figurarán las posiciones de l que forman s (en orden, y correspondientes siempre a la primera aparicin, si hay ms de una). En caso contrario, r será falso e is permanecerá sin cambios. Por ejemplo:

- $l = [1, 5, 7, 2, 1]$, $s = [1, 5, 2]$. En este caso, r debe ser verdadero e $is = [0, 1, 3]$. Notar que $[4, 1, 3]$ no es una solución válida, ya que 4 no es la primera aparición de 1 en l.
- $l = [\,'a','d','j','c'\,]$, $s = [\,]$. En este caso, r también debe ser verdadero (ya que $[\,]$ está incluida en l) e $is = [\,]$.
- $l = [1, 2, 3]$, $s = [3, 1]$. En este caso, r debe ser verdadero e $is = [2, 0]$.
- $l = [1, 2, 3]$, $s = [4]$. En este caso, r debe ser falso e is mantiene su valor original.