PROGRAMACIÓN 2. Curso 2017-18. Tercera prueba voluntaria de evaluación. Grupo 411

Esta es la tercera prueba de **evaluación voluntaria** que se plantea en la asignatura *Programación 2* a los alumnos matriculados en el grupo de mañanas. Debe ser resuelta individualmente.

Ejercicio 1 [10.0 puntos]

Considérese el siguiente procedimiento:

```
// Pre:  0 < n ≤ #v
// Post: nP = Num α ∈ [0, n − 1].(v[α] %2 = 0) ∧
//       sP = nP + ∑ α ∈ [0, n − 1].v[α] · (1 − v[α] %2)

void cosasDePares(const int v[], const int n, int &nP, int &sP) {
    int i = 0;
    nP = 0;
    sP = 0;
    while(i<n) {
        if (v[i] %2 == 0) {
            nP++;
            sP = sP+v[i];
        }
        i++;
    }
    sP = sP+nP;
}
```

Se deben resolver las siguientes cuestiones, y en este orden:

1. Definir un predicado I, invariante al bucle, que será el utilizado para llevar a cabo de verificación
2. Probar que I se verifica al llegar al bucle
3. Probar que I es suficientemente fuerte como para asegurar la evaluación de la guarda de la iteración
4. Probar que I es, efectivamente, invariante a la ejecución de una iteración del bucle
5. Definir una funcion de cota asociada al bucle
6. Probar que dicha función decrece estrictamente en cada iteración
7. Completar la verificación de la corrección, probando que al terminar la función se cumple la postcondición establecida

Se valorará la corrección, claridad y concisión de las respuestas. Si se considera que es necesario añadir aspectos a verificar, se deben indicar y llevar a cabo.

Una posible solución al ejercicio

1. Definir un predicado I, invariante al bucle, que será el utilizado para llevar a cabo de verificación Consideremos el siguiente predicado:

$$I \quad : \quad nP = Num\ \alpha \in [0, i-1].(v[\alpha]\ \%2 = 0)\ \wedge \tag{1}$$
$$sP = \sum\ \alpha \in [0, i-1].v[\alpha] \cdot (1 - v[\alpha]\ \%2)\ \wedge \tag{2}$$
$$0 \leq i \wedge i \leq n \wedge n \leq \#v \tag{3}$$

2. Probar que I se verifica al llegar al bucle. Para esto es suficiente probar que el siguiente código es correcto:

```
//Pre:  0 < n ≤ #v
    i = 0;
    nP = 0;
    sP = 0;
//I
```

Como el dominio de las tres instrucciones es *true* es suficiente probar que $Pre \longrightarrow ((I^0_{sP})^0_{nP})^0_i$. LLevando a cabo las sustituciones, nos lleva a tener que probar que

$$0 < n \leq \#v \quad \longrightarrow \quad 0 = Num\ \alpha \in [0, -1].(v[\alpha]\ \%2 = 0)\ \wedge \tag{4}$$
$$0 = \sum\ \alpha \in [0, -1].v[\alpha] \cdot (1 - v[\alpha]\ \%2)\ \wedge \tag{5}$$
$$0 \leq 0 \wedge 0 \leq n \tag{6}$$

La parte (4) se verifica pues el dominio vacío hace que el conteo sea 0. Esto, junto con que el sumatorio de (5) se haga sobre el dominio vacío hacen la segunda parte de la conjunción cierta. La parte (6) se deduce directamente de Pre.

3. Probar que I es suficientemente fuerte como para asegurar la evaluación de la guarda de la iteración. Se cumple trivialmente pues el dominio de la guarda del *while* es *true* (la guarda siempre se puede evaluar).

4. Probar que I es, efectivamente, invariante a la ejecución de una iteración del bucle. Para ello es suficiente con probar que la siguiente especificación es correcta:

```
//I ∧ i < n
    if (v[i] %2 == 0) {
        nP++;
        sP = sP+v[i];
    }
    i++;
//I
```

Para ello vamos a probar tres cosas:

La guarda se puede evaluar siempre: que es lo mismo que probar que $I \wedge\ i < n \longrightarrow Dom(v[i]\ \%2\ ==\ 0)$. Sustituyendo el dominio: $I \wedge\ i < n \longrightarrow 0 \leq i \wedge\ i < \#v$, que se deduce directamente de (3) y de $i < n$.

La parte *true* es correcta: que es lo mismo que probar la corrección de la siguiente especificación:

```
//I ∧ i < n ∧ v[i] %2 = 0
    nP++;
    sP = sP+v[i];
//I_i^{i+1}
```

Teniendo en cuenta que el dominio de ambas asignaciones es correcto (la única cuestión sería el rango de i en la segunda, pero ya lo hemos probado para asegurar la evaluación de la guarda del *if* y *nP++* no modifica el valor de i), es suficiente probar que $I \wedge\ i < n \wedge\ v[i]\,\%2 = 0 \longrightarrow ((I_i^{i+1})_{sP}^{sP+v[i]})_{nP}^{nP+1}$. La parte derecha se puede reescribir como

$$nP + 1 = Num\ \alpha \in [0, i].(v[\alpha]\,\%2 = 0)\ \wedge \tag{7}$$
$$sP + v[i] = \sum\ \alpha \in [0, i].v[\alpha] \cdot (1 - v[\alpha]\,\%2) \wedge \tag{8}$$
$$0 \leq i \wedge i \leq n \wedge n \leq \#v \tag{9}$$

Considerando $v[\alpha]\,\%2 = 0$ y (1) se puede concluir (7). Por otro lado, como $v[\alpha]\,\%2 = 0$, entonces $1 - v[\alpha]\,\%2 = 1$ que, junto con (2), permite concluir (8). Por último, (9) es lo mismo que (3).

La parte *false* es correcta: que es lo mismo que probar la corrección de la siguiente especificación:

```
//I ∧ i < n ∧ v[i] %2 ≠ 0

//I_i^{i+1}
```

Se trata de probar que $I \wedge\ i < n \wedge\ v[i]\,\%2 \neq 0 \longrightarrow I_i^{i+1}$. La parte derecha es

$$nP = Num\ \alpha \in [0, i].(v[\alpha]\,\%2 = 0)\ \wedge \tag{10}$$
$$sP = \sum\ \alpha \in [0, i].v[\alpha] \cdot (1 - v[\alpha]\,\%2) \wedge \tag{11}$$
$$0 \leq i \wedge i \leq n \wedge n \leq \#v \tag{12}$$

Considerando $v[\alpha]\,\%2 \neq 0$ y (1) se puede concluir (10). Por otro lado, como $v[\alpha]\,\%2 \neq 0$, entonces $1 - v[\alpha]\,\%2 = 0$ que, junto con (2), permite concluir (11). Por último, (12) es lo mismo que (3).

5. Definir una funcion de cota asociada al bucle. En cada iteración de bucle i crece, por lo que parece natural elegir $f(n, i) = n - i$ como función de cota. Nótese que $I \longrightarrow f(n, i) \geq 0$, por lo que es definida no negativa en el bucle.

6. Probar que dicha función decrece estrictamente en cada iteración. Sería suficiente probar que

```
//I ∧ i < n ∧ f(n,i) = K
    if (v[i] %2 == 0) {
        nP++;
        sP = sP+v[i];
    }
    i++;
//f(n,i) < K
```

Lo que es claro pues el valor de i se ha incrementado en una unidad. En cualquier caso, si se quiere hacer formalmente, y dado que el if no modifica el valor de la función f, sería suficiente probar que $I \wedge\ i < n \wedge\ f(n, i) = K \longrightarrow (f(n, i) < K)_i^{i+1}$, que es lo mismo que probar que $n - i = K \longrightarrow n - (i + 1) < K$, que es trivialmente cierto.

7. Completar la verificación de la corrección, probando que al terminar la función se cumple la postcondición establecida. Es suficiente con probar la corrección de la siguiente especificación

```
//I ∧ i ≮ n
    sP = sP+nP;
//Post
```

Como el dominio de la instrucción es *true*, basta probar que $I \wedge\ i \not< n \longrightarrow \text{Post}_{sP}^{sP+nP}$, siendo

$$\begin{aligned} \text{Post}_{sP}^{sP+nP} \quad = \quad & nP = Num\ \alpha \in [0, n-1].(v[\alpha]\,\%2 = 0)\ \wedge & (13) \\ & sP + nP = nP + \sum\ \alpha \in [0, n-1].v[\alpha] \cdot (1 - v[\alpha]\,\%2) & (14) \end{aligned}$$

Nótese que (3) junto con $i \not< n$ implican que $i = n$. Trivialmente, $(1) \wedge\ i = n \longrightarrow (13)$ y también $(2) \wedge\ i = n \longrightarrow (14)$.

Se valorará la corrección, claridad y concisión de las respuestas. Si se considera que es necesario añadir aspectos a verificar, se deben indicar y llevar a cabo.

Modo de entrega

Cada alumno entregará al profesor de su grupo un informe con las soluciones planteadas, en papel (ya sea manuscrito o impreso). No se admitirán trabajos presentados de otra forma, ni fuera de plazo o plagiados (total o parcialmente). El plazo de entrega termina el lunes 28 de mayo de 2018 a las 17 horas. Una vez publicadas las calificaciones, cada alumno podrá recoger su ejercicio corregido en el despacho del profesor.