DEPARTAMENTO DE COMPUTACIÓN
UNIVERSIDADE DA CORUÑA

Tecnología de la programación
Ingeniería Técnica en Informática de Sistemas. Curso 2006/07
Convocatoria Extraordinaria de Diciembre (21 de Diciembre de 2006)

Nombre : .....

Apellidos : .....

D.N.I. : .....

Instrucciones: En las preguntas 1 a 4 marcar todas las respuestas correctas. Cada pregunta correcta se valora con +0,5 puntos, incorrecta con -0,25, y no contestada con 0. Una puntuación total negativa en las preguntas 1 a 4 es valorada como 0 de cara al resto del examen.

Pregunta 1: Dado el programa $IF:$ **if** $b[i] = 0 \rightarrow$ skip **fi**, entonces:

×(a) $wp(IF, T)$ implica que $enrango(i, b)$

(b) equivale a **if** $b[i] = 0 \rightarrow$ skip $\mid b[i] \neq 0 \rightarrow b[i] := b[i]$ **fi**

(c) $wp(IF, R) \equiv R$

×(d) $wp(IF, T)$ implica que $b[i] = 0$

(a) **Sí**. Ya que en la definición de $wp(IF, R)$ para cualquier postcondición R se exige (entre otras cosas) $dominio(BB)$ que en este caso, conlleva $enrango(i, b)$.

(b) **No**. Ya que el programa original aborta cuando $b[i]$ no es cero (no existe rama para $b[i] \neq 0$), mientras que el propuesto, en ese caso, asigna el valor de $b[i]$ a $b[i]$ y finaliza correctamente.

(c) **No**. La definición general es:
$wp(IF, R) = dominio(BB)$ cand BB cand $(B_1 \Rightarrow wp(S_1, R)) \wedge ... \wedge (B_n \Rightarrow wp(S_n, R))$

que aplicado al ejemplo, se queda en: $enrango(i, b)$ cand $b[i] = 0$ cand $(b[i] = 0 \Rightarrow wp(skip, R))$

y ya que $wp(skip, R) \equiv R$, obtenemos $enrango(i, b)$ cand $b[i] = 0$ cand $(b[i] = 0 \Rightarrow R)$

que es equivalente a: $enrango(i, b)$ cand $b[i] = 0$ cand R

(d) **Sí**. Ya que en la definición de $wp(IF, R)$ para cualquier postcondición R se exige (entre otras cosas) que la disyunción de los guardianes sea cierta, lo que en este caso equivale a exigir $b[i] = 0$.

Pregunta 2: Dada la expresión $\alpha : 0 \leq h \leq k \leq n \wedge (\forall i \in [0, h-1] : b[i] \neq 0) \wedge (\forall i \in [k+1, n] : b[i] \neq 0)$ se cumple que:

(a) la variable k aparece ligada en α

×(b) las variables h y k aparecen libres en α

(c) la sustitución α^i_j es igual a $0 \leq h \leq k \leq n \wedge (\forall j \in [0, h-1] : b[j] \neq 0) \wedge (\forall j \in [k+1, n] : b[j] \neq 0)$

×(d) la sustitución α^k_m es igual a $0 \leq h \leq m \leq n \wedge (\forall j \in [0, h-1] : b[j] \neq 0) \wedge (\forall j \in [m+1, n] : b[j] \neq 0)$

(a) **No**. Aparece libre en $0 \leq h \leq k \leq n$ y en el rango de la variable i.

(b) **Sí**. Aparecen libres en $0 \leq h \leq k \leq n$ y en los rangos de la variable i respectivamente.

(c) **No**. Ya que la variable i aparece ligada a los dos cuantificadores.

(d) **Sí**. Las apariciones de la variable libre k han sido sustituidas por m. Además, la variable cuantificada i ha sido cambiada por otra variable (j), lo cual es independiente de cualquier sustitución.

Pregunta 3: El programa **do** $B_1 \rightarrow$ **if**$B_2 \rightarrow S_1 | \neg B_2 \rightarrow S_2$ **fi od** siempre equivale a:

(a) **do** $B_1 \wedge B_2 \rightarrow S_1 | B_1 \wedge \neg B_2 \rightarrow S_2$ **od**

×(b) **do** B_1 **cand** $B_2 \rightarrow S_1 | B_1$ **cand** $\neg B_2 \rightarrow S_2$ **od**

(c) **do** B_2 **cand** $B_1 \rightarrow S_1 | \neg B_2$ **cand** $B_1 \rightarrow S_2$ **od**

×(d) **do** $B_1 \rightarrow$ **if**$\neg B_2 \rightarrow S_2 | B_1 \rightarrow S_1$ **fi od**

(a) **No**. Porque podría darse el caso que B_2 fuese no definido. Como no se ha protegido B_1 con un **cand**, si B_1 es cierto ambas ramas devuelven no definido y el **do** aborta.

(b) **Sí**. Al añadir el **cand** se evita el problema que acabamos de mencionar.

(c) **No**. Porque ahora en el **cand** estamos evaluando primero la condición que puede devolver no definido.

(d) **No**. Porque las ramas del **if** son distintas a las del programa original.

Pregunta 4: Dado el programa $i, j, k := 0, 0, 0;$ **do** $i < 3 \rightarrow i, k := i + 1, k + 1; \mid j < 5 \rightarrow j, k := j + 1, k + 1$ **od**

(a) es equivalente a $i, j := 0, 0;$ **do** $i < 3 \vee j < 5 \rightarrow i, j, k := i + 1, j + 1, k + 1;$ **od**

(b) es determinista si partimos de un estado inicial en el que $i \geq 3$ o $j \geq 5$

×(c) es equivalente a $i, j, k := 3, 5, 8$

×(d) siempre es no determinista

(a) No. Para empezar, en el programa propuesto la k no se inicializa a cero. Pero lo que es más importante, mientras el programa original debe ejecutar las tres iteraciones para la i y las cinco para la j sin un orden preestablecido (esto es, de forma no determinista), en el programa propuesto, en cuanto j alcanza el valor 5, el bucle finaliza. Al constar de una única rama, el bucle propuesto ejecuta exactamente cinco iteraciones, con lo que la variable i termina también con valor 5 y la k se incrementa cinco veces.

(b) No. El valor inicial de i, j o k es irrelevante, ya que siempre se comienza inicializando esas variables a cero. Después, nada más comenzar el bucle tenemos que tanto $i < 3$ como $j < 5$ son ciertos, con lo que el programa debe ejecutar una de las dos ramas de modo no determinista. Esta elección no determinista se repetirá hasta que una de las dos variables alcance su límite (a partir de ahí, el programa ya sólo puede ejecutar la rama que incrementa la otra variable). De este modo, el programa siempre se enfrenta a alguna elección no determinista a lo largo de su ejecución, sin importar el estado inicial.

(c) Sí. Puesto que el bucle debe ejecutar las 3 iteraciones de i y las 5 de j, k se incrementa en todas ellas, es decir $3 + 5 = 8$ veces. Es necesario destacar que aunque el programa inicial era un bucle y el propuesto es una asignación, para comprobar su equivalencia tan sólo se tiene en cuenta el estado final que alcanzan.

(d) Si. La explicación es la misma que la de la segunda respuesta.

Pregunta 5: (2 puntos) Calcula y simplifica: $wp(\text{"}b[j], b[b[k]] := b[j], b[k]; b[j], b[k] := k, b[j]\text{"}, b[k] = b[b[k]])$.

$wp(\text{"}b[j], b[b[k]] := b[j], b[k]; b[j], b[k] := k, b[j]\text{"}, b[k] = b[b[k]])$

$\equiv wp(\text{"}b[j], b[b[k]] := b[j], b[k]\text{"}, \underbrace{wp(\text{"}b[j], b[k] := k, b[j]\text{"}, b[k] = b[b[k]])})$

Calculamos primero el wp para la última instrucción:

$wp(\text{"}b[j], b[k] := k, b[j]\text{"}, b[k] = b[b[k]]) \equiv enrango(j, b) \text{cand } enrango(k, b) \text{ cand } (b[k] = b[b[k]])^b_{(b;j:k;k:b[j])}$

Desarrollando el último término obtenemos:

$\equiv (b; j : k; k : b[j])[k] = (b; j : k; k : b[j])[(b; j : k; k : b[j])[k]] \equiv b[j] = (b; j : k; k : b[j])[b[j]]$

$\equiv (k = b[j] \wedge b[j] = b[j]) \vee (k \neq b[j] \wedge b[j] = (b; j : k)[b[j]])$

$\equiv k = b[j] \vee (k \neq b[j] \wedge b[j] = (b; j : k)[b[j]])$

$\equiv k = b[j] \vee b[j] = (b; j : k)[b[j]]$

$\equiv k = b[j] \vee (j = b[j] \wedge b[j] = k) \vee (j \neq b[j] \wedge b[j] = b[b[j]])$ (simplificamos $\alpha \vee (\beta \wedge \alpha) \equiv \alpha$)

$\equiv k = b[j] \vee (j \neq b[j] \wedge b[j] = b[b[j]])$

Sustituyendo ahora en la expresión inicial:

$wp(\text{"}b[j], b[b[k]] := b[j], b[k]\text{"}, enrango(j, b) \text{ cand } enrango(k, b) \text{ cand } (k = b[j] \vee (j \neq b[j] \wedge b[j] = b[b[j]])))$

$\equiv enrango(j, b) \text{ cand } enrango(k, b) \text{ cand } (k = b[j] \vee (j \neq b[j] \wedge b[j] = b[b[j]]))^b_{(b;j:b[j];b[k]:b[k])}$
$\equiv enrango(j, b) \text{ cand } enrango(k, b) \text{ cand } enrango(b[k], b) \text{ cand } (k = b[j] \vee (j \neq b[j] \wedge b[j] = b[b[j]]))^b_{(b;j:b[j];b[k]:b[k])}$

Resolviendo el último término:

$(k = b[j] \vee (j \neq b[j] \wedge b[j] = b[b[j]]))^b_{(b;j:b[j];b[k]:b[k])}$

$k = (b; j : b[j]; b[k] : b[k])[j]$
$\vee (j \neq (b; j : b[j]; b[k] : b[k])[j] \wedge (b; j : b[j]; b[k] : b[k])[j] = (b; j : b[j]; b[k] : b[k])[(b; j : b[j]; b[k] : b[k])[j]])$

$(b[k] = j \wedge (k = b[k] \vee (j \neq b[k] \wedge b[k] = (b; j : b[j]; b[k] : b[k])[b[k]])))$
$\vee (b[k] \neq j \wedge (k = (b; j : b[j])[j] \vee (j \neq (b; j : b[j])[j] \wedge (b; j : b[j])[j] = (b; j : b[j]; b[k] : b[k])[(b; j : b[j])[j]])))$

$(b[k] = j \wedge (k = b[k] \vee (j \neq b[k] \wedge b[k] = b[k])))$
$\vee (b[k] \neq j \wedge (k = b[j] \vee (j \neq b[j] \wedge b[j] = (b; j : b[j]; b[k] : b[k])[b[j]])))$

$(b[k] = j \wedge (k = b[k] \vee j \neq b[k]))$
$\vee (b[k] \neq j \wedge (k = b[j] \vee (j \neq b[j] \wedge b[j] = (b; j : b[j]; b[k] : b[k])[b[j]])))$

$(b[k] = j \wedge k = b[k])$
$\vee (b[k] \neq j \wedge k = b[j]) \vee \underbrace{(b[k] \neq j \wedge j \neq b[j] \wedge b[j] = (b; j : b[j]; b[k] : b[k])[b[j]])}_{\alpha}$

Resolvemos el término α:

$\alpha \equiv (b[k] \neq j \wedge j \neq b[j] \wedge b[j] = (b; j : b[j]; b[k] : b[k])[b[j]])$

$(b[k] = b[j] \wedge b[k] \neq j \wedge j \neq b[j] \wedge b[j] = b[k])$
$\vee (b[k] \neq b[j] \wedge b[k] \neq j \wedge j \neq b[j] \wedge b[j] = (b; j : b[j])[b[j]])$

$(b[k] = b[j] \wedge b[k] \neq j \wedge j \neq b[j])$
$\vee (b[k] \neq b[j] \wedge b[k] \neq j \wedge j \neq b[j] \wedge b[j] = b[b[j]])$

Simplificando mediante $(a \wedge b) \vee (\neg a \wedge b \wedge c) \equiv (b \wedge a) \vee (b \wedge c)$, tenemos:

$(b[k] = b[j] \wedge b[k] \neq j \wedge j \neq b[j])$
$\vee (b[k] \neq j \wedge j \neq b[j] \wedge b[j] = b[b[j]])$

Sustituimos α en la expresión de partida:

$(b[k] = j \wedge k = b[k]) \vee (b[k] \neq j \wedge k = b[j]) \vee (b[k] = b[j] \wedge b[k] \neq j \wedge j \neq b[j])$
$\vee (b[k] \neq j \wedge j \neq b[j] \wedge b[j] = b[b[j]])$

$(b[k] = j \wedge k = b[k]) \vee (b[k] \neq j \wedge k = b[j]) \vee (b[k] = b[j] \wedge b[k] \neq j)$
$\vee (b[k] \neq j \wedge j \neq b[j] \wedge b[j] = b[b[j]])$

Con lo que la solución final quedaría:

$\equiv enrango(j,b)$ cand $enrango(k,b)$ cand $enrango(b[k],b)$ cand $(b[k]=j \wedge k=b[k]) \vee (b[k]\neq j \wedge k=b[j])$
$b[k]=b[j] \wedge b[k]\neq j) \vee (b[k]\neq j \wedge j\neq b[j] \wedge b[j]=b[b[j]])$

Pregunta 6: (4 puntos) Dado un array $b[0:n-1]:integer$ con $n>0$, se desea escribir un programa que devuelva un booleano que indique si la suma de los elementos del vector es cero y las sumas parciales hasta cada uno de los elementos (incluido) es mayor o igual a cero. Se pide:

6.1 Establecer una precondición y una postcondición adecuadas

$\{Q:\ b[0:n-1]:integer \wedge n>0 \wedge b=B\}$
$\{R:\ e=((\sum j: 0\leq j<n: b[j])=0 \wedge (\forall k: 0\leq k<n: (\sum l: 0\leq l\leq k: b[l])\geq 0)) \wedge b=B\}$

6.2 Fijar una invariante y una función cota

$\{P:\ 0\leq i\leq n \wedge e=((\sum j: 0\leq j<i: b[j])=c \wedge (\forall k: 0\leq k<i: (\sum l: 0\leq l\leq k: b[l])\geq 0)) \wedge b=B\}$
$\{t:\ n-i\}$

6.3 Escribir el programa

$S_0: i,c,e:=0,0,T;$
do $B_1: i\neq n$ **cand** $(c+b[i])\geq 0 \rightarrow S_1: i,c:=i+1,c+b[i];$
od
if $i=n \wedge c=0 \rightarrow e:=T;$
$\mid (i=n \wedge c\neq 0) \vee (i\neq n) \rightarrow e:=F;$
fi

6.4 Demostrar su corrección *total*

a) $Q \Rightarrow wp(S_0,P)$

El wp resultante sería:

$0\leq 0\leq n \wedge T=((\sum j: 0\leq j<0: b[j])=0 \wedge (\forall k: 0\leq k<0: (\sum l: 0\leq l\leq k: b[l])\geq 0)) \wedge b=B$

En donde:

$(\sum j: 0\leq j<0: b[j])=0$ es cierto porque el sumatorio de un rango vacío es igual a cero.

El *para todo* de un rango vacío también es cierto, por lo tanto:

$\equiv 0\leq n \wedge T \equiv 0\leq n$, que se deduce directamente de Q.

b) $\{P\wedge B_i\}S_i\{P\}$ para $1\leq i\leq n$.

En este caso particular:

$P\wedge B \Rightarrow wp(S_1,P) \equiv P \wedge (i\neq n \text{ cand } (c+b[i])\geq 0 \Rightarrow wp(\text{"}i,c:=i+1,c+b[i]\text{"},P)$

El wp resultante sería:

$$\equiv 0\leq i+1\leq n \wedge e=(\overbrace{(\sum j: 0\leq j<i+1: b[j])=c+b[i]}^{a} \wedge \overbrace{(\forall k: 0\leq k<i+1: (\sum l: 0\leq l\leq k: b[l])\geq 0)}^{b}) \wedge b=B$$

Para la parte $0 \leq i+1 \leq n$ basta con tener en cuenta que $0 \leq i \leq n$ aparece en P mientras que B establece que $i \neq n$. La última parte de la conjunción, $b = B$, aparece directamente en P. Y para demostrar la segunda parte de la conjunción, es necesario separar el caso i en cada cuantificador:

(a) $(\sum j : 0 \leq j < i+1 : b[j]) = c + b[i]$

En P tenemos: $(\sum j : 0 \leq j < i : b[j]) = c$.

Sustituyendo en la parte derecha de la implicación y separando el caso i tenemos:

$(\sum j : 0 \leq j < i : b[j]) + b[i] = c + b[i] \equiv c + b[i] = c + b[i] \equiv T$

(b) $(\forall k : 0 \leq k < i+1 : (\sum l : 0 \leq l \leq k : b[l]) \geq 0)$

Separamos el caso $k = i$: $(\forall k : 0 \leq k < i : (\sum l : 0 \leq l \leq k : b[l]) \geq 0) \wedge \underbrace{(\sum l : 0 \leq l \leq i : b[l]) \geq 0}_{\alpha}$

La primera parte de la conjunción es cierta por P. En α volvemos a separar el caso $l = i$:

$((\sum l : 0 \leq l < i : b[l]) + b[i]) \geq 0 \equiv c + b[i] \geq 0 \equiv T$ por P y B.

c) $P \wedge \neg B \Rightarrow R^{'}$ con $R^{'} = Q_{IF}$

$R^{'} = b = B \wedge (i = n \wedge e = ((\sum j : 0 \leq j < i : b[j]) = c \wedge (\forall k : 0 \leq k < i : (\sum l : 0 \leq l \leq k : b[l]) \geq 0))$
$\vee (i < n \wedge e = ((\sum j : 0 \leq j < i : b[j]) = c \wedge (\exists k : 0 \leq k \leq i : (\sum l : 0 \leq l \leq k : b[l]) < 0)))$

$0 \leq i \leq n \wedge e = ((\sum j : 0 \leq j < i : b[j]) = c \wedge (\forall k : 0 \leq k < i : (\sum l : 0 \leq l \leq k : b[l]) \geq 0)) \wedge b = B$
$\wedge (i = n \text{ cor } (c + b[i]) < 0) \Rightarrow$

$$b = B \wedge \overbrace{(i = n \wedge e = ((\sum j : 0 \leq j < i : b[j]) = c \wedge (\forall k : 0 \leq k < i : (\sum l : 0 \leq l \leq k : b[l]) \geq 0)))}^{\alpha}$$
$$\vee \underbrace{(i < n \wedge e = ((\sum j : 0 \leq j < i : b[j]) = c \wedge (\exists k : 0 \leq k \leq i : (\sum l : 0 \leq l \leq k : b[l]) < 0)))}_{\beta}$$

$b = B$ aparece directamente en P, quedando por demostrar la verdad tanto de α como de β.

En la parte izquierda de la implicación pueden darse dos casos:

(a) $P \wedge i = n$ que nos permite obtener directamente α.

(b) $P \wedge (c + b[i]) < 0$
Vamos a dividir β en dos partes:

(b.1) $(\sum j : 0 \leq j < i : b[j]) = c$ que es cierto por P.

(b.2) $(\exists k : 0 \leq k \leq i : (\sum l : 0 \leq l \leq k : b[l]) < 0))$

Descomponemos la expresión para separar el caso $k = i$:

$(\exists k : 0 \leq k < i : (\sum l : 0 \leq l \leq k : b[l]) < 0)) \vee ((\sum l : 0 \leq l \leq i : b[l]) < 0)$
$(\exists k : 0 \leq k < i : (\sum l : 0 \leq l \leq k : b[l]) < 0)) \vee (((\sum l : 0 \leq l < i : b[l]) + b[i]) < 0)$

Por P, podemos sustituir $(\sum l : 0 \leq l < i : b[l])$ por c:

$(\exists k : 0 \leq k < i : (\sum l : 0 \leq l \leq k : b[l]) < 0)) \vee ((c + b[i]) < 0)$

El segundo término de la disyunción es cierto por $\neg B$, y por lo tanto la expresión β es cierta.

d) $P \wedge B \Rightarrow (t > 0)$

En P tenemos que $i \leq n$ que junto con $B \equiv i \neq n$ conlleva $i < n$, que a su vez equivale a $n - i > 0$.

e) $\{P \wedge B\}t_1 := t; S\{t < t_1\}$.

$P \wedge (i \neq n \text{ cand } (c + b[i]) \geq 0)) \Rightarrow wp(\text{"}t_1 := n - i; i, c := i + 1, c + b[i]\text{"}, n - i < t_1)$
$P \wedge (i \neq n \text{ cand } (c + b[i]) \geq 0)) \Rightarrow ((n - i < t_1)^{i,c}_{i+1,c+b[i]})^{t_1}_{n-i}$
$P \wedge (i \neq n \text{ cand } (c + b[i]) \geq 0)) \Rightarrow (n - (i + 1) < t_1)^{t_1}_{n-i}$
$P \wedge (i \neq n \text{ cand } (c + b[i]) \geq 0)) \Rightarrow n - (i + 1) < n - 1$
$P \wedge (i \neq n \text{ cand } (c + b[i]) \geq 0)) \Rightarrow -1 < 0.$
que es cierto.

de esta forma queda demostrada la corrección total de la estructura repetitiva.
Nos quedaría por demostrar la corrección de la estructura alternativa. Ello implica demostrar:

a) $Q \Rightarrow BB$

$Q \Rightarrow (i = n \wedge c = 0) \vee ((i = n \wedge c \neq 0) \vee (i \neq n))$
$Q \Rightarrow (i = n \wedge (c = 0 \vee c \neq 0)) \vee (i \neq n))$
$Q \Rightarrow (i = n \wedge T) \vee (i \neq n))$
$Q \Rightarrow (i = n \vee i \neq n)$
$Q \Rightarrow T$
Por lo tanto la implicación es cierta.

b) $Q \wedge B_i \Rightarrow wp(S_i, R)$. Tenemos que distinguir dos casos:

(2.a) $Q \wedge B_1 \Rightarrow wp(S_1, R) \equiv Q \wedge (i = n \wedge c = 0) \Rightarrow wp(\text{"}e := T\text{"}, R)$

$Q \wedge (i = n \wedge c = 0) \Rightarrow R^e_T$

Por B_1 en Q sólo puede ser cierto:

$b = B \wedge (i = n \wedge (\sum j : 0 \leq j < i : b[j]) = c \wedge (\forall k : 0 \leq k < i : (\sum l : 0 \leq l \leq k : b[l]) \geq 0))$

que con $B_1 : c = 0$ nos permite obtener directamente R.

(2.b) $Q \wedge B_2 \Rightarrow wp(S_2, R) \equiv Q \wedge ((i = n \wedge c \neq 0) \vee (i \neq n)) \Rightarrow wp(\text{"}e := F\text{"}, R)$

De nuevo tenemos que distinguir dos casos:

(2.b.1) En Q es cierto: $(i = n \wedge (\sum j : 0 \leq j < i : b[j]) = c \wedge (\forall k : 0 \leq k < i : (\sum l : 0 \leq l \leq k : b[l]) \geq 0))$
y en B_2 es cierto: $(i = n \wedge c \neq 0)$.

En este caso:

$R^e_F \equiv F = ((\sum j : 0 \leq j < n : b[j]) = 0 \wedge (\forall k : 0 \leq k < n : (\sum l : 0 \leq l \leq k : b[l]) \geq 0))$
$\equiv F = F \wedge (\forall k : 0 \leq k < n : (\sum l : 0 \leq l \leq k : b[l]) \geq 0)$
$\equiv F = F$ que es cierto.

(2.b.2) En Q es cierto: $(i < n \wedge e = ((\sum j : 0 \leq j < i : b[j]) = c \wedge (\exists k : 0 \leq k \leq i : (\sum l : 0 \leq l \leq k : b[l]) < 0)))$
y en B_2 es cierto: $(i \neq n)$.

$R^e_F \equiv F = ((\sum j : 0 \leq j < i : b[j]) = 0 \wedge (\forall k : 0 \leq k < n : (\sum l : 0 \leq l \leq k : b[l]) \geq 0)))$
$\equiv F = ((\sum j : 0 \leq j < i : b[j]) = 0 \wedge F)$
$\equiv F = F$ que es cierto.

Pregunta 7: (1 punto) Dado un array $b[0:n-1]:integer$ con $n>0$ y que contiene al menos un cero, se desea calcular en j y k las posiciones del primer y del último cero, respectivamente. Si hay un único cero en b entonces el programa devolverá $j=k=$ posición del cero. Establecer la precondición y la postcondición, la invariante, el programa y demostrar la corrección *parcial* (**equivale al ejercicio obligatorio de demostración formal**).

La especificación formal sería:

$\{Q: b[0:n-1]:integer \wedge n>0 \wedge (\exists i \in [0,n\rangle : b[i]=0)\}$

$\{R: 0 \leq j \leq k < n \wedge b[j]=0 \wedge b[0:j-1] \neq 0 \wedge b[k]=0 \wedge b[k+1,n-1] \neq 0\}$

Dado que no existe interferencia entre la búsqueda de j y la de k, la opción más sencilla es la de construir dos bucles independientes, cada uno con una parte de la postcondición. La precondición Q puede ser común, dado que el array b no se modifica en ningún caso. Por ejemplo, para el primer bucle fijamos la postcondición:

$\{R_1: 0 \leq j < n \wedge b[j]=0 \wedge b[0:j-1] \neq 0\}$

Para este caso la invariante resulta de suprimir $b[j]=0$ en R_1:

$\{P_1: 0 \leq j < n \wedge b[0:j-1] \neq 0\}$

y el programa es un simple bucle donde BB será $\neg(b[j]=0)$, es decir:

$j:=0;$
do $b[j] \neq 0 \rightarrow j:=j+1;$ **od**

Los tres pasos de la corrección parcial serían los siguientes:

1. $Q \Rightarrow wp(\text{"}j:=0\text{"}, P_1)$

El wp resultante es:
$0 \leq 0 < n \wedge \underbrace{b[\overbrace{0:-1}^{\emptyset}] \neq 0}_{T} \equiv 0 < n$ que se deduce directamente de Q.

2. $P_1 \wedge b[j] \neq 0 \Rightarrow wp(\text{"}j:=j+1\text{"}, P_1)$

El wp resultante es:
$0 \leq j+1 < n \wedge b[0:j] \neq 0$
$\equiv -1 \leq j < n-1 \wedge \underbrace{b[0:j-1] \neq 0}_{P_1} \wedge \underbrace{b[j] \neq 0}_{B_1}$

La condición $-1 \leq j$ se deduce de $0 \leq j$ en P_1. Por otro lado, también tenemos $j<n$ en P_1. Ahora bien, si tuviésemos $j=n-1$, la fórmula $b[0:j-1] \neq 0$ en P_1 junto con $B1$ querrían decir que no existen ceros en todo el array b lo que contradice la precondición Q (que se debe mantener cierta, pues b no se modifica). Por tanto, obligatoriamente, $j \neq n-1$, lo que implica $j<n-1$.

3. $P_1 \wedge b[j]=0 \Rightarrow R$

Trivial ya que P_1 se obtuvo eliminando $b[j]=0$ de R.

La segunda parte del ejercicio es análoga a la anterior. Podemos tomar como precondición $Q \wedge R_1$ y como postcondición e invariante las siguientes:

$\{R_2 : 0 \leq k < n \wedge b[k] = 0 \wedge b[k+1 : n-1] \neq 0\}$

$\{P_2 : 0 \leq k < n \wedge b[k+1 : n-1] \neq 0\}$

El bucle correspondiente es ahora:

$k := n - 1;$
do $b[k] \neq 0 \rightarrow k := k - 1;$ **oD**

Y los pasos de prueba:

1. $Q \Rightarrow wp(\text{"}k := n - 1\text{"}, P_2)$

El wp resultante es:

$0 \leq n - 1 < n \wedge \underbrace{b\overbrace{[n-1+1 : n-1]}^{\emptyset} \neq 0}_{T} \equiv 1 \leq n \equiv 0 < n$ que se deduce directamente de Q.

2. $P_2 \wedge b[k] \neq 0 \Rightarrow wp(\text{"}k := k - 1\text{"}, P_2)$

El wp resultante es:
$0 \leq k - 1 < n \wedge b[k : n-1] \neq 0 \equiv 1 \leq k < n + 1 \wedge \underbrace{b[k+1 : n-1] \neq 0}_{P_2} \wedge \underbrace{b[k] \neq 0}_{B_2}$

La condición $k < n + 1$ se deduce de $k < n$ en P_2. Para obtener $1 \leq k$ el razonamiento es similar al del bucle anterior ($k = 0$ junto con P_2 y $B2$ implicaría que no existen ceros en todo el array b lo que contradice la precondición Q (que se debe mantener cierta, pues b no se modifica).

3. $P_2 \wedge b[k] = 0 \Rightarrow R_2$

Trivial ya que P_2 se obtuvo eliminando $b[k] = 0$ de R.

Restaría tan sólo demostrar que la R final se satisface. Pero esto es sencillo, dado que j no se modifica durante el bucle para k, en realidad podemos demostrar que $R_1 \wedge R_2$ son ciertos al finalizar ambos bucles. Por otro lado, si $j > k$, es fácil ver que $R_1 \wedge R_2$ se vuelve contradictorio. Por tanto $j \leq k$ y obtenemos directamente que $R_1 \wedge R_2 \Rightarrow R$.