Comenzado el	jueves, 17 de octubre de 2024, 12:13
Estado	Finalizado
Finalizado en	jueves, 17 de octubre de 2024, 12:16
Tiempo empleado	3 minutos 10 segundos
Calificación	**-1,17** de 10,00 (**-11,67**%)

Pregunta 1

Sin contestar

Se puntúa como 0 sobre 1,00

Dada la siguiente especificación:

$\{0 \leq n \leq longitud\,(v)\ \wedge \forall z: 0 \leq z < n: v[z] \geq 0\}$

fun parejasParImpar(int v[ ], int n) dev (int p)

$\{p = \#i,j: 0 \leq i < j < n: \big(v[i]\%2 = 0 \wedge v\big[j\big]\%2 = 1\big)\}$

se ha decidido diseñar un bucle con coste $O\,(n)$ para resolverlo usando el siguiente invariante

$\{\ 0 \leq k \leq n \wedge p = \big(\#i,j: 0 \leq i < j < k: v[i]\%2 = 0 \wedge v\big[j\big]\%2 = 1\big)\ \wedge npares = \#l: 0 \leq l < k: v[l]\%2 = 0\}$

Indica cuáles de los siguientes bucles es correcto con respecto a la especificación y tiene dicho invariante

1.

```
int npares = 0, k=0; p=0;
while (k<n){
  if (v[k]%2==0) {npares = npares+1;}
  else {p = p + npares;}
  k=k+1;
}
return p;
```

2.

```
int npares = 0, k=0; p=0;
while (k<n){
  k=k+1;
  if (v[k-1]%2==0) {npares = npares+1;}
  else {p = p + npares;}
}
return p;
```

3.

```
int npares = 0, k=0; p=0;
while (k<n){
  if (v[k]%2==0) {npares = npares+1;}
  p = p + npares;
  k=k+1;
}
return p;
```

Seleccione una:

- a. Solamente 1 y 2.
- b. Solamente 1.
- c. Solamente 1 y 3.
- d. Ninguna de las anteriores.

a. Cierto. Los algoritmos 1 y 2 son correctos y su invariante es el indicado, aunque el cuerpo de los bucles es distinto porque se restablecen las variables en distinto orden. El algoritmo 3 no es correcto: por ejemplo si el vector tuviese los valores [6,2] , el algoritmo devolvería 3 cuando debería devolver 0, ya que no hay ningún impar.
b. Falso. El algoritmo 2 también es correcto y tiene el mismo invariante que 1, pero restablece las variables del bucle en distinto orden.
c. Falso. El algoritmo 3 no es correcto: por ejemplo si el vector tuviese los valores [6,2] , el algoritmo devolvería 3 cuando debería devolver 0, ya que no hay ningún impar.
d. Falso. La respuesta correcta es: Solamente 1 y 2.

La respuesta correcta es: Solamente 1 y 2.

Pregunta 2

Sin contestar

Se puntúa como 0 sobre 1,00

Dada la siguiente especificación

$\{v.size(\,) > 0\}$

fun ultimoMaximo(vector v) dev (int i)

$\{0 \leq i < v.size(\,) \wedge (\forall k: 0 \leq k < v.size(\,): v[k] \leq v[i]\,) \wedge (\forall l: i < l < v.size(\,): v[l] < v[i]\,)\}$

y el siguiente algoritmo como cuerpo de la función

```
i=v.size()-1; int j=v.size()-2;
while (j >= 0){
  if (v[j]>v[i]) {i=j;}
  j=j-1;
}
return i;
```

indica si el algoritmo es correcto con respecto a la especificación y en tal caso cual es el invariante que permite demostrar la corrección del bucle.

Seleccione una:

a. Es correcto con invariante $-1 \leq j < i < v.size(\,) \wedge \big(\forall k: j < k < v.size(\,): v[k] \leq v[i]\big) \wedge (\forall l: i < l < v.size(\,): v[l] < v[i]\,)$.

b. Es correcto con invariante $0 \leq j < i < v.size(\,) \wedge \big(\forall k: j < k < v.size(\,): v[k] \leq v[i]\big) \wedge (\forall l: i < l < v.size(\,): v[l] < v[i]\,)$.

c. Es correcto con invariante $-1 \leq j < i < v.size(\,) \wedge \big(\forall k: j \leq k < v.size(\,): v[k] \leq v[i]\big) \wedge (\forall l: i < l < v.size(\,): v[l] < v[i]\,)$.

d. Ninguna de las anteriores.

a. Cierto.
b. Falso. El último valor que toma j es -1, así que ha de ser $-1 \leq j < i < v.size(\,)$.
c. Falso. Este invariante está mal definido cuando $j = -1$.
d. Falso. La respuesta correcta es: Es correcto con invariante
$-1 \leq j < i < v.size(\,) \wedge \big(\forall k: j < k < v.size(\,): v[k] \leq v[i]\big) \wedge (\forall l: i < l < v.size(\,): v[l] < v[i]\,)$.

La respuesta correcta es: Es correcto con invariante
$-1 \leq j < i < v.size(\,) \wedge \big(\forall k: j < k < v.size(\,): v[k] \leq v[i]\big) \wedge (\forall l: i < l < v.size(\,): v[l] < v[i]\,)$.

Pregunta 3

Sin contestar

Se puntúa como 0 sobre 1,00

Dada la siguiente especificación

$\{v.size\,(\,) > 0\}$

fun primerMaximo(vector v) dev (int i)

$\{0 \leq i < v.size\,(\,) \wedge (\forall k: 0 \leq k < v.size\,(\,) : v[k] \leq v[i]\,) \wedge (\forall l: 0 \leq l < i: v[l] < v[i]\,)\}$

y el siguiente algoritmo como cuerpo de la función

```
i=0; int j=1;
while (j < v.size()){
  if (v[j]>v[i]) {i=j;}
  j=j+1;
}
return i;
```

indica si el algoritmo es correcto con respecto a la especificación y en tal caso cuál es el invariante que permite demostrar la corrección del bucle.

Seleccione una:

- a. Es correcto con invariante $0 \leq i < j \leq v.size\,(\,) \wedge \big(\forall k: 0 \leq k < j: v[k] \leq v[i]\big) \wedge (\forall l: 0 \leq l < i: v[l] < v[i]\,)$.
- b. Es correcto con invariante $0 \leq i < j \leq v.size\,(\,) \wedge (\forall k: 0 \leq k < i: v[k] \leq v[i]\,) \wedge (\forall l: 0 \leq l < i: v[l] < v[i]\,)$.
- c. Es correcto con invariante $0 \leq i < j < v.size\,(\,) \wedge \big(\forall k: 0 \leq k < j: v[k] \leq v[i]\big) \wedge (\forall l: 0 \leq l < i: v[l] < v[i]\,)$.
- d. Ninguna de las anteriores.

a. Cierto.
b. Falso. Para obtener la postcondición, necesitamos que el elemento v[i] sea mayor o igual que todos los del vector, no solo de los que están a su izquierda, por lo que debería ser $(\forall k: 0 \leq k < j: v[k] \leq v[i])$.
c. Falso. El último valor de j es v.size() por lo que debería ser $0 \leq i < j \leq v.size\,(\,)$.
d. Falso. La respuesta correcta es: Es correcto con invariante $0 \leq i < j \leq v.size\,(\,) \wedge \big(\forall k: 0 \leq k < j: v[k] \leq v[i]\big) \wedge (\forall l: 0 \leq l < i: v[l] < v[i]\,)$.

La respuesta correcta es: Es correcto con invariante $0 \leq i < j \leq v.size\,(\,) \wedge \big(\forall k: 0 \leq k < j: v[k] \leq v[i]\big) \wedge (\forall l: 0 \leq l < i: v[l] < v[i]\,)$.

Pregunta 4

Sin contestar

Se puntúa como 0 sobre 1,00

Dada la siguiente especificación

$\{0 \leq n \leq longitud\,(v)\}$

fun numProd (int v[ ], int n) dev (int l)

$\{l = \#i\colon 0 \leq i < n\colon v[i] = \big(\Pi j\colon 0 \leq j < i\colon v\big[j\big]\big)\}$

Se ha escrito el siguiente fragmento de código como cuerpo de dicha función:

```
int k=0, p=1; l=0;
while (k<n)
{
 p=p*v[k];
 if (p==v[k+1]) {l=l+1;}
 k=k+1;
}
return l;
```

Indica si este algoritmo es correcto con respecto a la especificación y en tal caso cual es el invariante que permite demostrar la corrección del bucle.

Seleccione una:

- a. Es correcto con invariante $\{0 \leq k \leq n \wedge p = (\Pi z\colon 0 \leq z \leq k\colon v[z]) \wedge l = \#i\colon 0 \leq i \leq k\colon v[i] = \big(\Pi j\colon 0 \leq j < i\colon v\big[j\big]\big)\}$
- b. Es correcto con invariante $\{0 \leq k < n \wedge p = (\Pi z\colon 0 \leq z < k\colon v[z]) \wedge l = \#i\colon 0 \leq i < k\colon v[i] = \big(\Pi j\colon 0 \leq j < i\colon v\big[j\big]\big)\}$
- c. El algoritmo no es correcto con respecto a la especificación.
- d. Ninguna de las anteriores.

a. Falso. Con ese invariante la variable p debería inicializarse con $v[0]$, no con 1, pero dicho acceso sería erróneo en caso de que n fuese 0. Análogamente la variable l debería tomar inicialmente el valor 1 si $v[0] = 1$ y 0 si $v[0] \neq 1$.
b. Falso. La variable k puede llegar a tomar el valor n. Además, la variable l se actualiza de forma incorrecta, debería compararse p con $v[k]$ no con $v[k+1]$.
c. Cierto. El algoritmo no es correcto, intenta acceder a la posición n del vector, que está fuera de rango.
d. Falso. La respuesta correcta es: El algoritmo no es correcto con respecto a la especificación.

La respuesta correcta es: El algoritmo no es correcto con respecto a la especificación.

Pregunta 5

Sin contestar

Se puntúa como 0 sobre 1,00

Dada la siguiente especificación:

$\{0 < n \leq longitud\,(a)\}$

fun maximo(int a[ ], int n) dev (int m)

$\{m = \max i: 0 \leq i < n: a[i]\}$

y el siguiente invariante:

$\{0 \leq k < n \wedge m = \max i: k \leq i < n: a[i]\}$

indica qué expresiones colocar en lugar de los ? para que el siguiente algoritmo sea correcto con respecto a la especificación dada y el bucle tenga como invariante el indicado:

```
k=? ; m=?;
while (?)
{
  if (m<a[k-1]) {m=a[k-1];}
  k=k-1;
}
```

Seleccione una:

- a. Inicialización: k=n;m=a[n]; Condición del while: (k>0)
- b. Inicialización: k=n;m=a[n-1]; Condición del while: (k>1)
- c. Inicialización: k=n-1;m=a[n-1]; Condición del while: (k>0)
- d. Ninguna de las anteriores.

a. Falso. Según el invariante k no puede tomar el valor n.
b. Falso. Según el invariante k no puede tomar el valor n.
c. Cierto. Si k vale $n-1$ inicialmente, el máximo está bien definido, siendo $i = n-1$ el único índice en el rango, por lo que m ha de valer $a[n-1]$.
d. Falso. La respuesta correcta es: Inicialización: k=n-1;m=a[n-1]; Condición del while: (k>0)

La respuesta correcta es: Inicialización: k=n-1;m=a[n-1]; Condición del while: (k>0)

Pregunta 6

Incorrecta

Se puntúa -0,33 sobre 1,00

Indica cuál de las siguientes es una función de cota para este bucle:

```
int i=0; bool encontrado=false;
while (i<n){
 if (v[i]>0)
    {encontrado=true;}
 else
    {i=i+1;}
}
```

Seleccione una:

- a. Ninguna de ellas.
- b. $i+1$
- c. $n-i+1$ ✖ Incorrecta. La variable i no siempre se incrementa, por lo que $n-i+1$ no decrece en todas las vueltas del bucle.
- d. i

a. Correcta. Si hay un valor >0 en el vector el bucle no termina.
b. Incorrecta. La variable i no decrece en cada vuelta.
c. Incorrecta. La variable i no siempre se incrementa, por lo que $n-i+1$ no decrece en todas las vueltas del bucle.
d. Incorrecta. La variable i no decrece en cada vuelta.

La respuesta correcta es: Ninguna de ellas.

Pregunta 7

Incorrecta

Se puntúa -0,50 sobre 1,00

La precondición más débil en:

{??}

```
int x=0;
int y=1/x;
```

{**true**}

es **false**.

Seleccione una:

- a. Verdadero
- b. Falso ✖

Verdadero. Para que la segunda asignación esté bien definida es necesario que $x \neq 0$, pero la instrucción anterior le asigna el valor 0. Al hacer la sustitución, el predicado $0 \neq 0$ equivale a **false**.

La respuesta correcta es: Verdadero

Pregunta 8

Sin contestar

Se puntúa como 0 sobre 1,00

Dada la siguiente especificación

$\{v.size(\,) > 0\}$

fun ultimoMaximo(vector v) dev (int i)

$\{0 \leq i < v.size(\,) \wedge (\forall k: 0 \leq k < v.size(\,): v[k] \leq v[i]\,) \wedge (\forall l: i < l < v.size(\,): v[l] < v[i]\,)\}$

y el siguiente algoritmo como cuerpo de la función

```
i=v.size()-1; int j=v.size()-2;
while (j >= 0){
  if (v[j]>v[i]) {i=j;}
  j=j-1;
}
return i;
```

indica si el algoritmo es correcto con respecto a la especificación y en tal caso cual es el invariante que permite demostrar la corrección del bucle.

Seleccione una:

a. Es correcto con invariante $0 \leq j < i < v.size(\,) \wedge \big(\forall k: j < k < v.size(\,): v[k] \leq v[i]\big) \wedge (\forall l: i < l < v.size(\,): v[l] < v[i]\,)$.

b. Es correcto con invariante $-1 \leq j < i < v.size(\,) \wedge \big(\forall k: j \leq k < v.size(\,): v[k] \leq v[i]\big) \wedge (\forall l: i < l < v.size(\,): v[l] < v[i]\,)$.

c. El algoritmo no es correcto con respecto a la especificación.

d. Ninguna de las anteriores.

a. Falso. El último valor que toma j es -1, así que ha de ser $-1 \leq j < i < v.size(\,)$.
b. Falso. Este invariante está mal definido cuando $j = -1$.
c. Falso. El algoritmo es correcto con invariante
$-1 \leq j < i < v.size(\,) \wedge \big(\forall k: j < k < v.size(\,): v[k] \leq v[i]\big) \wedge (\forall l: i < l < v.size(\,): v[l] < v[i]\,)$.
d. Cierto. La respuesta correcta es: Es correcto con invariante
$-1 \leq j < i < v.size(\,) \wedge \big(\forall k: j < k < v.size(\,): v[k] \leq v[i]\big) \wedge (\forall l: i < l < v.size(\,): v[l] < v[i]\,)$.

La respuesta correcta es: Ninguna de las anteriores.

Pregunta 9

Incorrecta

Se puntúa -0,33 sobre 1,00

Indica cuál es la precondición más débil:

{??}

```
x = y;
y = x;
```

$\{x = 3 \land y = 4\}$

Seleccione una:

a. **true** ✖ Falso. Al sustituir en la postcondición y por x y luego x por y se obtiene $y = 3 \land y = 4$, así que la precondición más débil es **false**.

b. **false**

c. $y = 3 \land x = 4$

d. Ninguna de las anteriores.

a. Falso. Al sustituir en la postcondición y por x y luego x por y se obtiene $y = 3 \land y = 4$, así que la precondición más débil es **false**.
b. Cierto. Al sustituir en la postcondición y por x y luego x por y se obtiene $y = 3 \land y = 4$.
c. Falso. Al sustituir en la postcondición y por x y luego x por y se obtiene $y = 3 \land y = 4$, así que la precondición más débil es **false**.
d. Falso. La respuesta correcta es: **false**

La respuesta correcta es: **false**

Pregunta 10

Sin contestar

Se puntúa como 0 sobre 1,00

Dada la siguiente especificación:

$\{0 \leq n \leq longitud\,(a)\}$

fun contarPares(int a[ ], int n) dev (int c)

$\{c = \#i\colon 0 \leq i < n\colon a[i]\%2 = 0\}$

y el siguiente algoritmo:

```
int contarPares(int a[],int n){
 int c=0; int k=-1;
 while (k<n-1)
 {
  if (a[k+1] % 2 == 0) {c=c+1;}
  k=k+1;
 }
 return c;
}
```

indica si el algoritmo es correcto con respecto a la especificación y en tal caso cual es el invariante que permite demostrar la corrección del bucle.

Seleccione una:

- a. Es correcto con invariante $\{-1 \leq k < n \wedge c = \#i\colon 0 \leq i \leq k\colon a[i]\%2 = 0\}$
- b. Es correcto con invariante $\{-1 \leq k \leq n \wedge c = \#i\colon 0 \leq i < k\colon a[i]\%2 = 0\}$.
- c. Es correcto con invariante $\{-1 \leq k \leq n \wedge c = \#i\colon 0 \leq i < n\colon a[i]\%2 = 0\}$.
- d. Ninguna de las anteriores.

a. Cierto.
b. Falso. k nunca llega a tomar el valor n, empieza con valor -1 y termina en $n-1$, así que el invariante correcto es $\{-1 \leq k < n \wedge c = \#i\colon 0 \leq i \leq k\colon a[i]\%2 = 0\}$.
c. Falso. c lleva el número de pares hasta k incluida, solamente cuando ha recorrido todo el vector tiene el número de pares de todo el vector. Así que el invariante es $\{-1 \leq k < n \wedge c = \#i\colon 0 \leq i \leq k\colon a[i]\%2 = 0\}$.
d. Falso. La respuesta correcta es: Es correcto con invariante $\{-1 \leq k < n \wedge c = \#i\colon 0 \leq i \leq k\colon a[i]\%2 = 0\}$

La respuesta correcta es: Es correcto con invariante $\{-1 \leq k < n \wedge c = \#i\colon 0 \leq i \leq k\colon a[i]\%2 = 0\}$