Cuestiones Algoritmos Iterativos

Fecha de Creación	@9 de octubre de 2023 19:34
Asignatura	FAL
Fecha de Modificación	@16 de octubre de 2023 15:14

Pregunta 1
Parcialmente correcta
Se puntúa 0,17 sobre 1,00
Marcar pregunta

Determina cuáles de las siguientes afirmaciones son ciertas:

Seleccione una o más de una:

- a. Cada bucle tiene únicamente un invariante.
- b. **True** es invariante de cualquier bucle.
- c. Para que un invariante pueda usarse para justificar la corrección de un bucle, dicho invariante debe ser suficientemente fuerte. ✔ **Cierto**. El invariante debe ser lo suficientemente fuerte como para permitir que, una vez que se falsifica la condición del bucle, pueda justificarse que (posiblemente tras algunas instrucciones de finalización) se satisface la postcondición.
- d. Si un predicado es un invariante de un bucle, entonces permite justificar su corrección. ✖ **Falso**. El invariante debe ser suficientemente fuerte para, entre otras cosas, permitir que, una vez que se falsifica la condición del bucle, pueda justificarse que (posiblemente tras algunas instrucciones de finalización) se satisface la postcondición.
- e. **False** puede ser invariante de algún bucle.

a. **Falso**. Un bucle puede mantener invariantes muchos predicados (por ejemplo, el bucle en `int i=0; while(int i<n) {i++;}` tiene como invariantes a **true**, $0 \leq i \leq n, i \geq 0$ ...
b. **Cierto**. **True** siempre es cierto, haga lo que haga el algoritmo. Por tanto, será cierto al comienzo del bucle, y también tras cada iteración.
c. **Cierto**. El invariante debe ser lo suficientemente fuerte como para permitir que, una vez que se falsifica la condición del bucle, pueda justificarse que (posiblemente tras algunas instrucciones de finalización) se satisface la postcondición.
d. **Falso**. El invariante debe ser suficientemente fuerte para, entre otras cosas, permitir que, una vez que se falsifica la condición del bucle, pueda justificarse que (posiblemente tras algunas instrucciones de finalización) se satisface la postcondición.
e. **Falso**. **False** siempre es falso, haga lo que haga el algoritmo. Por tanto, será falso incluso antes de la primera iteración.

Las respuestas correctas son: **True** es invariante de cualquier bucle., Para que un invariante pueda usarse para justificar la corrección de un bucle, dicho invariante debe ser suficientemente fuerte.

Pregunta 2
Incorrecta
Se puntúa 0,00 sobre 1,00
Marcar pregunta

Considera el siguiente algoritmo:

$P \equiv \{0 \leq n \leq tam(a) \land k > 0\}$

```
int /* resul*/ incognita(int a[], int n, int k) {
    int resul = 0;
    int i = 0;
    int lon = 0;
    while (i < n) {
        if (a[i] == 0) {
            i++;
            lon++;
            if (lon == k) {
                resul++;
                lon--;
            }
        }
        else {
            i++;
            lon = 0;
        }
    }
    return resul;
}
```

$Q \equiv \{resul = \# i, j : 0 \leq i \leq j < n \land (\forall u : i \leq u \leq j : a[u] = 0) : (j - i) + 1 = k\}$

¿Cuáles de las siguientes afirmaciones son ciertas?

Seleccione una o más de una:

- a. $\forall u : i - lon \leq u \leq i : a[u] = 0$ es un invariante del bucle. ✖ **Falso**. Informalmente, afirma que, terminando en la posición i, hay un tramo de $lon + 1$ ceros. Pero esto no tiene porque ser necesariamente cierto, ya que, al comienzo de cada iteración, $a[i]$ no tiene porque ser 0.
- b. $\forall u : i - lon \leq u < i : a[u] = 0$ es un invariante del bucle.
- c. $resul = \# i, j : 0 \leq i \leq j < n \land (\forall u : i \leq u \leq j : a[u] = 0) : (j - i) + 1 = k$ es un invariante del bucle.
- d. $lon < k$ es un invariante del bucle. ✔ **Cierto**. Inmediatamente antes de la primera iteración, $lon = 0$. Como, por la precondición, $k > 0$, el predicado se verifica en este punto. Por otra parte, si se supone que, antes de comenzar a ejecutar el bucle, $lon < k$, puede observarse que, tras ejecutar el cuerpo del bucle, lon continuará siendo menor que k.
- e. $lon \leq k$ es un invariante del bucle.

a. **Falso**. Informalmente, afirma que, terminando en la posición i, hay un tramo de $lon + 1$ ceros. Pero esto no tiene porque ser necesariamente cierto, ya que, al comienzo de cada iteración, $a[i]$ no tiene porque ser 0.
b. **Cierto**. Informalmente, afirma que, inmediatamente a la izquierda de la posición i, hay un tramo con lon ceros. La invarianza de este hecho puede comprobarse examinando detalladamente el código.
c. **Falso**. Por ejemplo, antes de la primera iteración $resul = 0$, pero $\# i, j : 0 \leq i \leq j < n \wedge (\forall u : i \leq u \leq j : a[u] = 0) : (j - i) + 1 = k$ no tiene porque ser necesariamente 0. Lo que se está afirmando aquí es que la postcondición es, directamente, el invariante. Si esto fuera así, ¡ni siquiera sería necesario ejecutar el bucle! (tendriamos ya la solución inmediatamente después de realizar la inicialización de variables).
d. **Cierto**. Inmediatamente antes de la primera iteración, $lon = 0$. Como, por la precondición, $k > 0$, el predicado se verifica en este punto. Por otra parte, si se supone que, antes de comenzar a ejecutar el bucle, $lon < k$, puede observarse que, tras ejecutar el cuerpo del bucle, lon continuará siendo menor que k.
e. **Cierto**, ya que $lon < k$ es un invariante, y $lon < k \Rightarrow lon \leq k$.

Las respuestas correctas son: $\forall u : i - lon \leq u < i : a[u] = 0$ es un invariante del bucle.
, $lon < k$ es un invariante del bucle.
, $lon \leq k$ es un invariante del bucle.

Pregunta 3
Incorrecta
Se puntúa 0,00 sobre 1,00
Marcar pregunta

Considera el siguiente algoritmo:

$P \equiv \{0 \leq n \leq tam(a)\}$

```
bool /* resul*/ incognita(int a[], int n) {
      int i =n-2;
      bool resul=true;
      while(i >= 0 && resul) {
         if(a[i] >= a[i+1]) {
           resul = false;
         }
         i--;
      }
     return resul;
}
```

$Q \equiv \{resul = \forall i : 0 \leq i < n - 1 : a[i] < a[i+1]\}$

¿Cuáles de las siguientes afirmaciones son ciertas?

Seleccione una o más de una:

- [ ] a. $resul = \forall u : 0 \leq u < i - 1 : a[u] < a[u+1]$ es un invariante del bucle del algoritmo.
- [x] b. $resul = \forall u : i < u < n - 1 : a[u] < a[u+1]$ es un invariante del bucle del algoritmo. ✔ **Cierto**. Tras la inicialización de las variables, $(resul = \forall u : i < u < n - 1 : a[u] < a[u+1]) \equiv (true = \forall u : n - 2 < u < n - 1 : a[u] < a[u+1]) \equiv (true = \forall u : false$
Asimismo, analizando el código puede verse que, inmediatamente antes de comenzar a ejecutar el bucle, las indexaciones a[i] y a[i+1] t
deducirse a partir de la información del predicado!). Entonces, es inmediato comprobar que las acciones que se realizan tanto cuando a
preservan el predicado.
- [ ] c. $resul = \forall u : i < u < n - 1 : a[u] < a[u+1]$ es un invariante del bucle del algoritmo, pero no permite justificar su corrección parcial.
- [ ] d. $-2 \leq i < n - 1$ es un invariante del bucle del algoritmo.
- [x] e. $0 \leq i < n - 1$ es un invariante del bucle del algoritmo. ✘ **Falso**. De hecho, si $n = 0$, tras la inicialización $i = -2$, por lo que el predicado no se satisface ni siquiera antes de la primera iteración.

a. **Falso**. Basta considerar, por ejemplo, el vector $(1, 1, 1, 1)$. Tras la inicialización de las variables $(resul = \forall u : 0 \leq u < i - 1 : a[u] < a[u+1]) \equiv (true = \forall u : 0 \leq u < 1 : a[u] < a[u+1]) \equiv (true = (a[0] < a[1])) \equiv (true = false) \equiv false$. Por tanto, ni siquiera puede garantizarse que el predicado se satisfaga inmediatamente antes de la primera iteración del bucle.
b. **Cierto**. Tras la inicialización de las variables, $(resul = \forall u : i < u < n - 1 : a[u] < a[u+1]) \equiv (true = \forall u : n - 2 < u < n - 1 : a[u] < a[u+1]) \equiv (true = \forall u : false : a[u] < a[u+1])) \equiv (true = true) \equiv true$. Asimismo, analizando el código puede verse que, inmediatamente antes de comenzar a ejecutar el bucle, las indexaciones a[i] y a[i+1] tienen sentido (¡aunque este hecho no puede deducirse a partir de la información del predicado!). Entonces, es inmediato comprobar que las acciones que se realizan tanto cuando $a[i] < a[i+1]$ como cuando $a[i] \geq a[i+1]$ preservan el predicado.
c. **Cierto**. Aunque es un invariante, no permite justificar que las indexaciones en el cuerpo del bucle tienen sentido, ni tampoco determinar el valor de i a la salida del bucle.
d. **Cierto**. Tras la inicialización, $i = n - 2$. Por tanto, $i < n - 1$. Además, como, por la precondición, $n \geq 0$, $i \geq -2$. Por otra parte, si suponemos que, antes de ejecutar el cuerpo del bucle, se verifica $-2 \leq i < n - 1$, tras decrementar i seguirá verificandose $i < n - 1$. Por otra parte, la condición del bucle asegura que, antes de ejecutar el cuerpo, $i \geq 0$. Por tanto, tras decrementar i, se podrá asegurar que $i \geq -1 > -2$.
e. **Falso**. De hecho, si $n = 0$, tras la inicialización $i = -2$, por lo que el predicado no se satisface ni siquiera antes de la primera iteración.

Las respuestas correctas son: $resul = \forall u : i < u < n - 1 : a[u] < a[u+1]$ es un invariante del bucle del algoritmo.
, $resul = \forall u : i < u < n - 1 : a[u] < a[u+1]$ es un invariante del bucle del algoritmo, pero no permite justificar su corrección parcial.
, $-2 \leq i < n - 1$ es un invariante del bucle del algoritmo.

Pregunta 4
Incorrecta
Se puntúa 0,00 sobre 1,00
Marcar pregunta

Considera el siguiente algoritmo:

$P \equiv \{0 \leq n \leq tam(a)\}$

```
int /* resul*/ incognita(int a[], int n) {
      int i =0;
      int resul=0;
      while(i < n) {
         if(a[i]%2 ==0) {
            resul++;
            i++;
         }
      }
     return resul;
}
```

$Q \equiv \{resul = \#i : 0 \leq i < n : a[i]\%2 = 0\}$

¿Cuáles de las siguientes afirmaciones son ciertas?:

Seleccione una o más de una:

- a. La precondición puede hacerse suficientemente fuerte como para que el algoritmo sea correcto.
- b. El algoritmo no termina. ✘ **Falso**. La terminación o no terminación del algoritmo depende del contenido del vector. En particular, si el vector sólo contiene números pares, el algoritmo termina.
- c. La terminación del algoritmo depende del contenido del vector **int a[n]**. ✔ **Cierto**. Si el vector sólo contiene números pares, el algoritmo termina.
- d. El algoritmo es correcto.
- e. El algoritmo es parcialmente correcto.

a. **Cierto**. Por ejemplo, $P \equiv \{0 \leq n \leq tam(a) \wedge \forall i : 0 \leq i < n : a[i]\%2 = 0\}$ garantizará que el vector sólo contenga números pares, y, por tanto, que el algoritmo siempre termine.
b. **Falso**. La terminación o no terminación del algoritmo depende del contenido del vector. En particular, si el vector sólo contiene números pares, el algoritmo termina.
c. **Cierto**. Si el vector sólo contiene números pares, el algoritmo termina.
d. **Falso**. Si el vector tiene algún número impar, el algoritmo no termina, ya que, en este caso, el cuerpo del bucle no hace nada (recuérdese que, para que un algoritmo sea correcto, aparte de ser parcialmente correcto, debe terminar para todas sus entradas).
e. **Cierto**. Cuando el algoritmo termina, en $resul$ se han contabilizado el número de pares del vector, que es lo que indica la postcondición.

Las respuestas correctas son: La precondición puede hacerse suficientemente fuerte como para que el algoritmo sea correcto., La terminación del algoritmo depende del contenido del vector **int a[n]**., El algoritmo es parcialmente correcto.

Pregunta 5
Incorrecta
Se puntúa 0,00 sobre 1,00
Marcar pregunta

Considera el siguiente algoritmo:

$P \equiv \{0 < n \leq tam(a)\}$

```
int /* resul*/ incognita(int a[], int n) {
      int i =1;
      int resul=a[0];
      while(i < n) {
        if (a[i] > resul) {
           resul = a[i];
        }
        i++;
      }
     return resul;
}
```

$Q \equiv \{resul = max\ i : 0 \leq i < n : a[i]\}$

¿Cuáles de los siguientes predicados son invariantes del bucle que permiten justificar la corrección parcial del algoritmo?

Seleccione una o más de una:

- a. $resul = max\ u : 0 \leq u < i : a[u] \wedge 0 \leq i \leq n$.
- b. $resul = max\ u : 0 \leq u < i : a[u] \wedge i \leq n$.
- c. $resul = max\ u : 0 \leq u < i : a[u] \wedge 0 < i \leq n$.
- d. $resul = max\ u : 0 \leq u < i : a[u] \wedge 0 \leq i < n$. ✘ **Falso**. El predicado ni siquiera es invariante, ya que no lo es el término $0 \leq i < n$ (puede incumplirse, por ejemplo, incluso antes de comenzar a ejecutarse el bucle si $n = 1$).
- e. $resul = max\ u : 0 \leq u < i : a[u]$.

a. **Cierto**. Puede comprobarse que el predicado es, efectivamente, un invariante (obsérvese que, aunque i nunca es 0, $0 < i \Rightarrow 0 \leq i$, por lo que el término $0 \leq i \leq n$ es un invariante, aunque podría haberse elegido uno más fuerte, como $0 < i \leq n$). Además, al salir del bucle, la negación de la condición y este invariante permiten deducir que $i = n$, y, por tanto, la postcondición.
b. **Falso**. Aunque el predicado es un invariante, es demasiado débil. Efectivamente, aunque, cuando la condición del bucle se falsifica, es posible deducir la postcondición, el predicado no permite determinar que, en el cuerpo del bucle,la indexación $a[i]$ tiene sentido, porque no puede determinarse que $i \geq 0$.
c. **Cierto**. Puede comprobarse que el predicado es, efectivamente, un invariante. Además, al salir del bucle, la negación de la condición y este invariante permiten deducir que $i = n$, y, por tanto, la postcondición.
d. **Falso**. El predicado ni siquiera es invariante, ya que no lo es el término $0 \leq i < n$ (puede incumplirse, por ejemplo, incluso antes de comenzar a ejecutarse el bucle si $n = 1$).
e. **Falso**. Aunque el predicado es un invariante, es demasiado débil (se necesita información adicional para poder determinar que $i = n$ a la salida del bucle, así como para poder garantizar que las indexaciones a elementos del vector están dentro de rango).

Las respuestas correctas son: $resul = max\ u : 0 \leq u < i : a[u] \wedge 0 \leq i \leq n$.
, $resul = max\ u : 0 \leq u < i : a[u] \wedge 0 < i \leq n$.

Pregunta 6
Incorrecta
Se puntúa 0,00 sobre 1,00
Marcar pregunta

Considera el siguiente algoritmo:

```
int /* resul*/ incognita(int a[], int n) {
      int i =0;
      int resul=0;
      while(i < n) {
      }
     return resul;
}
```

¿Cuáles de las siguientes afirmaciones son ciertas?

Seleccione una o más de una:

- a. El algoritmo es parcialmente correcto, independientemente de su especificación. ✘ **Falso**. Por ejemplo, basta hacer $P \equiv \{true\}$, $Q \equiv \{resul \neq 0\}$, y ejecutar el algoritmo con $n = 0$.
- b. El algoritmo es correcto, independientemente de su especificación.
- c. El algoritmo no es parcialmente correcto.
- d. El algoritmo no es correcto. ✘ **Falso**. Por ejemplo, basta hacer $P \equiv \{n = 0\}$ y $Q \equiv \{resul = 0\}$ para que el algoritmo sea correcto.
- e. La corrección del algoritmo depende de su especificación.

a. **Falso**. Por ejemplo, basta hacer $P \equiv \{true\}$, $Q \equiv \{resul \neq 0\}$, y ejecutar el algoritmo con $n = 0$.
b. **Falso**. Por ejemplo, basta hacer $P \equiv \{true\}$ y observar que, para cualquier $n > 0$, el algoritmo no termina.
c. **Falso**. Por ejemplo, basta hacer $Q \equiv \{resul = 0\}$.
d. **Falso**. Por ejemplo, basta hacer $P \equiv \{n = 0\}$ y $Q \equiv \{resul = 0\}$ para que el algoritmo sea correcto.
e. **Cierto**. Por ejemplo, si hacemos $P \equiv \{n = 0\}$ y $Q \equiv \{resul = 0\}$ el algoritmo será correcto. Sin embargo, si hacemos $P \equiv \{true\}$ el algoritmo no lo será, independientemente de su postcondición (ya que para cualquier $n > 0$, el algoritmo no termina)

La respuesta correcta es: La corrección del algoritmo depende de su especificación.

Pregunta 7
Parcialmente correcta
Se puntúa 0,33 sobre 1,00
Marcar pregunta

Considera el siguiente algoritmo:

$P \equiv \{0 \leq n \leq tam(a)\}$

```
int /* resul*/ incognita(int a[], int n) {
      int i =0;
      int prod=1;
      int resul=0;
      while(i < n) {
        prod *= a[i];
        resul += prod;
        i++;
      }
     return resul;
}
```

$Q \equiv \{resul = \Sigma i : 0 \leq i < n : (\Pi j : 0 \leq j < i : a[j])\}$

¿Cuáles de las siguientes afirmaciones son ciertas?

Seleccione una o más de una:

- a. $resul = \Sigma u : 0 \leq u < i : (\Pi j : 0 \leq j < u : a[j]) \wedge 0 \leq i \leq n$ es un invariante del bucle del algoritmo, y permite comprobar su corrección parcial.
- b. $prod = \Pi u : 0 \leq u < i : a[u]$ es un invariante del bucle del algoritmo.
- c. $resul = \Sigma u : 0 \leq u < i : (\Pi j : 0 \leq j < u : a[j])$ es un invariante del bucle del algoritmo. ✔ **Cierto**. Este predicado dice que, para un i dado, $resul$ vale $a[0] + a[0] * a[1] + \ldots + a[0] * a[1] * \ldots * a[i-1]$, es decir, que el problema ha sido ya resuelto para el segmento que queda a la izquierda de i. La invarianza de este hecho puede comprobarse examinando con detalle el algoritmo.
- d. $|\Pi i : 0 \leq i < n : a[i]| - |prod|$ es una expresión de cota que permite probar la terminación de este algoritmo ($|x|$ significa *el valor absoluto de* x).
- e. $resul = \Sigma u : 0 \leq u < i : (\Pi j : 0 \leq j < u : a[j]) \wedge 0 \leq i \leq n$ es un invariante del bucle del algoritmo, pero no permite comprobar su corrección parcial.

a. **Falso**. El problema de este invariante es que no dice nada acerca de lo que vale $prod$, por lo que no podrá justificarse, únicamente en base al contenido del mismo, que se preserva en cada iteración.
b. **Cierto**. Informalmente, este predicado expresa que en $prod$ está el producto de los elementos del segmento que queda a la izquierda de i. La invarianza de este hecho puede comprobarse examinando con cuidado el algoritmo.
c. **Cierto**. Este predicado dice que, para un i dado, $resul$ vale $a[0] + a[0] * a[1] + \ldots + a[0] * a[1] * \ldots * a[i-1]$, es decir, que el problema ha sido ya resuelto para el segmento que queda a la izquierda de i. La invarianza de este hecho puede comprobarse examinando con detalle el algoritmo.
d. **Falso**. No puede garantizarse que esta expresión decrezca en cada iteración, ya que el vector puede contener ceros.
e. **Cierto**. El problema de este invariante es que no dice nada acerca de lo que vale $prod$, por lo que no podrá razonarse, únicamente en base al contenido del mismo, que se preserva en cada iteración.

Las respuestas correctas son: $prod = \Pi u : 0 \leq u < i : a[u]$ es un invariante del bucle del algoritmo.
, $resul = \Sigma u : 0 \leq u < i : (\Pi j : 0 \leq j < u : a[j])$ es un invariante del bucle del algoritmo.
, $resul = \Sigma u : 0 \leq u < i : (\Pi j : 0 \leq j < u : a[j]) \wedge 0 \leq i \leq n$ es un invariante del bucle del algoritmo, pero no permite comprobar su corrección parcial.

Pregunta 8
Parcialmente correcta
Se puntúa 0,50 sobre 1,00
Desmarcar

Determina cuáles de las siguientes afirmaciones son ciertas:

Seleccione una o más de una:

- [x] a. Si la postcondición de un algoritmo es $Q \equiv \{true\}$, entonces es parcialmente correcto. ✔ **Cierto**, ya que, siempre que el algoritmo termina, la postcondición se satisface trivialmente.
- [ ] b. Si la postcondición de un algoritmo es $Q \equiv \{true\}$, entonces es correcto.
- [ ] c. Si la precondición de un algoritmo es $P \equiv \{false\}$, entonces es correcto.
- [ ] d. Si la precondición de un algoritmo es $P \equiv \{true\}$, entonces es correcto.
- [ ] e. Si la postcondición de un algoritmo es $Q \equiv \{false\}$, entonces no es correcto.

a. **Cierto**, ya que, siempre que el algoritmo termina, la postcondición se satisface trivialmente.
b. **Falso**. Puede ocurrir que el algoritmo no siempre termine (para que el algoritmo sea correcto, aparte de ser parcialmente correcto, debe terminar siempre).
c. **Cierto**. Para que el algoritmo fuera incorrecto, debería poderse encontrar una entrada para la cuál (i) bien el algoritmo no termina; o (ii) bien el algoritmo termina, pero no satisface su postcondición. Pero, dado que ninguna entrada satisface la precondición, no es posible encontrar tal entrada (ni ninguna otra: la precondición 'prohibe' que el algoritmo se ejecute). Por tanto, como el algoritmo no puede ser incorrecto, la única alternativa que nos queda es que sea correcto.
d. **Falso**. Que la precondición sea **true** no implica que el algoritmo termine siempre, y que el estado que se alcanza satisfaga su postcondición. Lo único que implica es que el algoritmo se podrá ejecutar para cualquier posible valor de sus entradas.
e. **Falso**. Aunque basta que el algoritmo se ejecute para que ya no sea correcto (ya que la postcondición no se va a satisfacer nunca), aún queda la posibilidad de que la especificación impida que el algoritmo se ejecute alguna vez (precondición **false**).

Las respuestas correctas son: Si la postcondición de un algoritmo es $Q \equiv \{true\}$, entonces es parcialmente correcto.
, Si la precondición de un algoritmo es $P \equiv \{false\}$, entonces es correcto.

Pregunta 9
Correcta
Se puntúa 1,00 sobre 1,00
Marcar pregunta

Considera el siguiente algoritmo:

$P \equiv \{0 \leq n \leq tam(a)\}$

```
int /* resul*/ incognita(int a[], int n) {
      int i =n-1;
      int resul=0;
      while(i >= 0) {
        resul += a[i];
        i--;
      }
     return resul;
}
```

$Q \equiv \{resul = \Sigma i : 0 \leq i < n : a[i]\}$

¿Cuáles de las siguientes afirmaciones son ciertas?:

Seleccione una o más de una:

- [ ] a. $n - i$ es una expresión de cota que permite probar la terminación del bucle de este algoritmo.
- [x] b. i es una expresión de cota que permite probar la terminación del bucle de este algoritmo. ✔ **Cierto**. Por una parte, el mínimo valor que alcanzará i es -1. Por otra parte, i decrece en cada iteración.
- [ ] c. $resul = \Sigma u : i < u < n : a[u]$ es invariante del bucle de este algoritmo, y permite demostrar su corrección parcial.
- [ ] d. $resul = \Sigma u : 0 \leq u < i : a[u]$ es invariante del bucle de este algoritmo.
- [x] e. $resul = \Sigma u : i < u < n : a[u]$ es invariante del bucle de este algoritmo, pero no permite demostrar su corrección parcial. ✔ **Cierto**. El predicado es un invariante del bucle. Para verlo, basta observar que, justo antes del inicio del bucle, se tiene que $i = n - 1$ y $resul = 0$. Por tanto, en este estado $(resul = \Sigma u : i < u < n : a[u]) \equiv (0 = \Sigma u : n - 1 < u < n : a[u]) \equiv (0 = \Sigma u : false : a[u]) \equiv (0 = 0) \equiv true$. Por su parte, supongamos que el predicado se cumple al comienzo de la ejecución del cuerpo del bucle. Tras ejecutar `resul += a[i]`, se cumplirá $resul = \Sigma u : i \leq u < n : a[u]$, por lo que, tras ejecutar `i--`, se cumplirá de nuevo $resul = \Sigma u : i < u < n : a[u]$. Sin embargo, no es suficientemente fuerte para demostrar la corrección parcial, ya que, una vez que se sale del bucle, es necesario poder determinar que $i = -1$, y el invariante no contiene información suficiente para ello.

a. **Falso**. Como i decrece en cada iteración, $n - i$ crece. Por tanto, no puede ser una expresión de cota.
b. **Cierto**. Por una parte, el mínimo valor que alcanzará i es -1. Por otra parte, i decrece en cada iteración.
c. **Falso**. Aunque el predicado es un invariante del bucle, no permite demostrar la corrección parcial, ya que no permite determinar que, cuando se sale del bucle, $i = -1$, hecho necesario para deducir la postcondición a partir de este predicado.
d. **Falso**. Basta observar que, tras la inicialización de i y de $resul$, se llega a un estado en el que $i = n - 1$ y $resul = 0$. En este estado, se tiene $(resul = \Sigma u : 0 \leq u < i : a[u]) \equiv (0 = \Sigma u : 0 \leq u < n - 1 : a[u])$ predicado que, en la mayoría de los casos, será falso (v.g., si el vector es $(1, 1)$, entonces $\Sigma u : 0 \leq u < n - 1 : a[u]) = \Sigma u : 0 \leq u < 1 : a[u]) = a[0] = 1 \neq 0$).
e. **Cierto**. El predicado es un invariante del bucle. Para verlo, basta observar que, justo antes del inicio del bucle, se tiene que $i = n - 1$ y $resul = 0$. Por tanto, en este estado $(resul = \Sigma u : i < u < n : a[u]) \equiv (0 = \Sigma u : n - 1 < u < n : a[u]) \equiv (0 = \Sigma u : false : a[u]) \equiv (0 = 0) \equiv true$. Por su parte, supongamos que el predicado se cumple al comienzo de la ejecución del cuerpo del bucle. Tras ejecutar `resul += a[i]`, se cumplirá $resul = \Sigma u : i \leq u < n : a[u]$, por lo que, tras ejecutar `i--`, se cumplirá de nuevo $resul = \Sigma u : i < u < n : a[u]$. Sin embargo, no es suficientemente fuerte para demostrar la corrección parcial, ya que, una vez que se sale del bucle, es necesario poder determinar que $i = -1$, y el invariante no contiene información suficiente para ello.

Las respuestas correctas son: i es una expresión de cota que permite probar la terminación del bucle de este algoritmo.
, $resul = \Sigma u : i < u < n : a[u]$ es invariante del bucle de este algoritmo, pero no permite demostrar su corrección parcial.

Pregunta **10**
Incorrecta
Se puntúa 0,00 sobre 1,00
Marcar pregunta

Considera el siguiente algoritmo:

$P \equiv \{0 \leq n \leq tam(a)\}$

```
int /* resul*/ incognita(int a[], int n) {
      int i =0;
      int resul=0;
      while(i < n) {
        resul += a[i];
        i++;
      }
     return resul;
}
```

$Q \equiv \{resul = \Sigma i : 0 \leq i < n : a[i]\}$

¿Cuáles de las siguientes afirmaciones son ciertas?:

Seleccione una o más de una:

- a. $-i$ es una expresión de cota que permite probar la terminación del bucle de este algoritmo. ✖ **Falso**. Aunque $-i$ decrece en cada iteración, no puede garantizarse que alcanza un mínimo. Por tanto, no es una expresión de cota del bucle (una expresión de cota, aparte de decrecer en cada iteración, debe permitir garantizar que no baja por debajo de cierto valor).
- b. $resul = \Sigma u : 0 \leq u < i : a[u]$ es invariante del bucle de este algoritmo, pero no permite demostrar su corrección parcial.
- c. $resul = \Sigma u : 0 \leq u < i : a[u]$ es invariante del bucle de este algoritmo, y permite demostrar su corrección parcial. ✖ **Falso**. Tras la salida del bucle, no podemos afirmar que $i = n$, hecho necesario para deducir la postcondición a partir del invariante en dicho punto.
- d. $resul = \Sigma u : 0 \leq u < i : a[u]$ es invariante del bucle de este algoritmo.
- e. $2n - i$ es una expresión de cota que permite probar la terminación del bucle de este algoritmo.

a. **Falso**. Aunque $-i$ decrece en cada iteración, no puede garantizarse que alcanza un mínimo. Por tanto, no es una expresión de cota del bucle (una expresión de cota, aparte de decrecer en cada iteración, debe permitir garantizar que no baja por debajo de cierto valor).
b. **Cierto**. Necesitamos información adicional que nos permita afirmar que $i = n$ a la salida del bucle, para poder deducir la postcondición a partir del invariante en dicho punto.
c. **Falso**. Tras la salida del bucle, no podemos afirmar que $i = n$, hecho necesario para deducir la postcondición a partir del invariante en dicho punto.
d. **Cierto**. Tras la inicialización de i y de $resul$, se llega a un estado en el que $i = 0$ y $resul = 0$. En este estado, $(resul = \Sigma u : 0 \leq u < i : a[u]) \equiv (0 = \Sigma u : 0 \leq u < 0 : a[u]) \equiv (0 = \Sigma u : false : a[u]) \equiv (0 = 0) \equiv true$. Por tanto, el predicado se cumple justo antes de la primera iteración. Igualmente, si, antes de comenzar a ejecutarse el cuerpo del bucle asumimos que se cumple el predicado, tras `resul += a[i]` se cumplirá $resul = \Sigma u : 0 \leq u \leq i : a[u]$, ya que se ha sumado $a[i]$. Por tanto, tras `i++` volverá a cumplirse $resul = \Sigma u : 0 \leq u < i : a[u]$, ya que i se ha incrementado en una unidad. Como resultado, el predicado se cumple también tras cada iteración.
e. **Cierto**. La expresión decrece en cada iteración, ya que, en cada iteración, i aumenta en una unidad. Por otra parte, como i valdrá, como máximo, n, $2n - i$ nunca se hará más pequeña que n (obsérvese que el valor mínimo alcanzable por la cota no tiene porque ser necesariamente 0; puede ser cualquier otro valor, siempre y cuando pueda garantizarse que la expresión no cae por debajo de dicho valor).

Las respuestas correctas son: $resul = \Sigma u : 0 \leq u < i : a[u]$ es invariante del bucle de este algoritmo, pero no permite demostrar su corrección parcial.
, $resul = \Sigma u : 0 \leq u < i : a[u]$ es invariante del bucle de este algoritmo.
, $2n - i$ es una expresión de cota que permite probar la terminación del bucle de este algoritmo.