



Nombre:

1. (1.5 pts) Especifica (incluyendo precondition y postcondición) las siguientes funciones:

a) La función búsqueda binaria, teniendo en cuenta que sólo se puede aplicar sobre vectores ordenados. La función ha de devolver la posición y un booleano que indique si se ha encontrado el elemento.

b) La función moda de un vector, es decir, una función que devuelve el valor que más veces se repite en el vector.

2. (1 pts) En el tipo abstracto "lista con punto de interes" (Lista_PI) escribe las ecuaciones para las operaciones *modificar* y *extender*. *Modificar* sustituye el elemento sobre el que está el punto de interés por un nuevo elemento, si el punto de interés está al final devuelve un error. *Extender* añade un elemento al final de la lista siempre que el punto de interés esté al final de la lista, si el punto de interés no está al final devuelve un error.

Operaciones:

modificar: lista_PI, elem \rightarrow lista_PI

extender: lista_PI, elem \rightarrow lista_PI

Ecuaciones:



Nombre:

3. (1.5 pts) Las ecuaciones de la operación **pre** son:

pre: arbol \rightarrow pila

$\forall p, p1, p2$: pila, $\forall x$: elem, $\forall a1, a2$: arbol

1) **pre**(a_nulo) \equiv p_nula

2) **pre**(plantar(x, a1, a2)) \equiv apilar(x, concatenar(**pre**(a1), **pre**(a2)))

Conocemos además la siguiente ecuación sobre concatenar:

3) **altura**(concatenar(p1, p2)) \equiv **altura**(p1) + **altura**(p2)

Utilizando las ecuaciones de los tipos **pila** y **arbol** y las ecuaciones anteriores demostrar el siguiente *teorema inductivo*:

$\forall a$: arbol

tam(a) \equiv **altura**(**pre**(a))

**Nombre:**

4. (4 ptos) Nos han dado la siguiente función diciéndonos que sirve para calcular el producto de a por b para a y b naturales:

```
int f(int a, int b)
{
    assert( a >= 0 && b >= 0 );
    int resul;

    if(a == 0)
        resul = 0;
    else
    {
        resul = f(a / 2, b * 2);
        resul = resul + (a % 2) * b;
    }
    return resul;
}
```

a) (2 ptos) Comprueba si es cierto o no. Es decir, verifica totalmente la función. Ten en cuenta que la división entera y el resto vienen definidos por la siguiente ecuación:

$$x = y * (x \text{ div } y) + x \% y$$

b) (1 ptos) Transforma la función a recursiva final mediante la técnica de desplegado y plegado.

c) (1 ptos) Transforma la función a iterativa y di cual es su invariante y su función de cota.

**Nombre:**

5. (2 pts) Dada la siguiente función iterativa, derivar las instrucciones de dentro del bucle que faltan. Suponer que las instrucciones que aparecen en la función ya están verificadas, así como la función de cota.

suma: pila \rightarrow entero

1) suma(p_nula) \equiv 0

2) suma(apilar(x, p) \equiv x + suma(p)

func suma (p:pila de enteros) dev s: entero

{Pre: P = p}

s := 0;

{Inv: suma(p) + s = suma(P); Dec: altura(p) }

***[\sim nula(p) \rightarrow ????**

|

{Post: s = suma(P) }