Tarea 1

Tony Santiago Montes Buitrago

Departamento de Ingeniería de Sistemas y Computación, Universidad de los Andes

18 de agosto de 2021

Enunciado

Describir los datos de entrada, los datos de salida y especificar en forma de predicado de lógica proposicional o lógica de primer orden la precondición y la postcondición para los siguientes problemas

1. Dado un número natural, encontrar su raiz cuadrada entera, la cual se define como el mayor número natural que es menor que la raíz cuadrada real del número

Entradas

Variable	Tipo	Descripción
n	nat	Número natural

Salidas

Variable	Tipo	Descripción
raizEntera	nat	Raíz cuadrada entera de n

Precondición

n pertenece a los naturales, pero como eso ya está especificado en las entradas, la precondición es verdadera

`true`

Postcondición

$$\sqrt{n} - 1 < raizEntera \leq \sqrt{n}$$

2. Dado un número natural, determinar si es un número primo

Entradas

Variable	Tipo	Descripción
n	nat	Número natural

Salidas

Variable	Tipo	Descripción
esPrimo	bool	Indicador de si n es primo

Precondición

`true`

Postcondición

esPrimo es `true` si n es mayor a 1 y solo es divisible entre 1 y entre sí mismo
$esPrimo \equiv n > 1 \wedge (\forall d : nat | d > 0 \wedge d | n : d = 1 \vee d = n)$

3. Dado un número primo, encontrar el siguiente número primo

Entradas

Variable	Tipo	Descripción
p	nat	Número primo

Salidas

Variable	Tipo	Descripción
proxP	nat	Próximo primo después de p

Definiendo la función $primo(n) \equiv n > 1 \wedge (\forall d : nat | d > 0 \wedge d | n : d = 1 \vee d = n)$
Precondición

$$primo(p)$$

Postcondición

$$primo(proxP) \wedge proxP > p$$

4. Dado un arreglo de números enteros, encontrar la posición en la que se encuentra el número par más grande del arreglo. Si el arreglo no tiene números pares, la respuesta debe ser -1

Entradas

Variable	Tipo	Descripción
lst	array[0,N) of int	Arreglo de N números enteros

Salidas

Variable	Tipo	Descripción
i	int	Posición de lst en la que está el par más grande

Precondición

`true`

Postcondición

Ningún número de la lista es divisible entre 2 y entonces $i = -1$; o $lst[i]$ es mayor o igual a todos los números divisibles entre 2 de la lista
$((\forall k : int | 0 \leq k < N : \neg\, lst[k]|2) \wedge i = -1) \vee (\forall k : int | 0 \leq k < N : lst[k]|2 \implies lst[i] > lst[k])$

5. Dado un conjunto de estudiantes ordenados por código y un rango de fechas, encontrar los estudiantes que cumplen años en el rango de fechas dado

Entradas

Variable	Tipo	Descripción
est	array[0,N) of Est	Arreglo de N estudiantes (Est)
fIni	int	fecha inicial del rango
fFin	int	fecha final del rango (rango abierto)

Salidas

Variable	Tipo	Descripción
cumple	array[0,M) of Est	Arreglo de M estudiantes que cumplen años en el rango de fechas

NOTA: La clase *Est* se modela con los atributos:
•*codigo* que retorna el código del estudiante
•$fechaCumpleanios$ que retorna el número de días desde el 1 de Enero hasta el cumpleaños (ej. 1 de Febrero = 32)

Precondición

Al avanzar por la lista *estudiantes* cada código de estudiante es mayor que el anterior
$(\forall k : int | 0 \leq k < N - 1 : est[k].codigo < est[k+1].codigo)$

Postcondición

Cada estudiante de la lista *cumple* tiene una fecha de cumpleaños mayor o igual a $fIni$ y menor a $fFin$
$(\forall k : int | 0 \leq k < M - 1 : fIni \leq cumple[k].fechaCumpleanios < fFin)$

6. Dado un conjunto de n cadenas, encontrar la cadena más corta tal que cada una de las cadenas de entrada sea una subcadena de la cadena de salida

Entradas

Variable	Tipo	Descripción
cad	array[0,n) of String	Arreglo de n cadenas (String)

Salidas

Variable	Tipo	Descripción
final	String	Cadena más corta que contenga como subcadena, cada cadena de *cad*

NOTA: La clase *String* se modela con el método:
•$.contains(param)$ que verifica si la cadena actual contiene la cadena dada por parámetro.
NOTA: La clase *String* se modela con el atributo:
•$length$ que indica la longitud de la cadena

Precondición

`true`

Postcondición

Cada cadena de *cad* está contenida en $final$ y la longitud de $final$ es menor a la suma de la longitud de todas las cadenas de *cad*.
$(\forall k : int | 0 \leq k < n : final.contains(cad[k])) \wedge final.length < (\sum j : int | 0 \leq j < n : cad[j].length)$