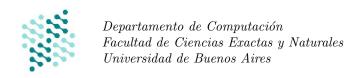
Algoritmos y Estructuras de Datos

Guía de laboratorio 2 Debugging, testing y manejo de archivos



Los ejercicios de *Debugging* y de *Manejo de Archivos* constituyen el Taller 2. Los ejercicios de Testing no se entregan. Para aprobar el taller, todos los tests que les proveemos deben pasar. Recuerden subir únicamente los archivos Debugging.java y Archivos.java al repositorio de entrega dentro de la carpeta taller2. Última fecha de entrega: domingo 03/09.

1. Debugging

Los siguientes ejercicios tienen ya un código que los implementa, aunque con algunos errores. Su objetivo es encontrar estos errores y arreglarlos para que pasen los tests.

Ejercicio 1. Debuggear el código que implementa el operador xor (que se simboliza con \veebar). Este operador representa un "o exclusivo": $A \veebar B$ es verdadero cuando A ó B es verdadero, pero no ambos. Cumple la siguiente tabla de verdad:

A	В	$A \veebar B$
false	false	false
false	true	true
${ m true}$	false	true
${ m true}$	true	false

Ejercicio 2. Debuggear el código que implementa la función iguales, que determina si dos arreglos de número enteros son iguales.

```
proc iguales (in xs: seq\langle\mathbb{Z}\rangle, in ys: seq\langle\mathbb{Z}\rangle) = res : Bool { requiere \{\text{true}\} asegura \{res=\text{true}\leftrightarrow |xs|=|ys|\wedge_L (\forall i:\mathbb{N})\ (i<|xs|\longrightarrow_L xs[i]=ys[i])\} }
```

Ejercicio 3. Debuggear el código que implementa la función todosPositivos, que determina si todos los valores de un arreglo de número enteros son positivos.

```
\begin{array}{l} \texttt{proc todosPositivos (in } xs : seq \langle \mathbb{Z} \rangle) = \texttt{res} : \texttt{Bool } \{ \\ \texttt{requiere } \{\texttt{true}\} \\ \texttt{asegura } \{res = \texttt{true} \leftrightarrow (\forall x : \mathbb{N}) \ (x \in xs \longrightarrow 0 < x) \} \\ \} \end{array}
```

Ejercicio 4. Debuggear el código que implementa la función maximo, que devuelve el máximo valor de un arreglo de números enteros

```
\begin{array}{l} \texttt{proc maximo (in } xs \colon seq\langle \mathbb{Z} \rangle) = \texttt{res} : \mathbb{Z} \  \, \{ \\ \quad \quad \texttt{requiere } \{|xs| > 0\} \\ \quad \quad \quad \texttt{asegura } \{res \in xs \land (\forall x : \mathbb{N}) \ (x \in xs \longrightarrow x \leq res)\} \\ \} \end{array}
```

Ejercicio 5. Debuggear el código que implementa la función ordenado, que determina si un arreglo de número enteros está ordenado de manera creciente.

```
proc ordenado (in xs: seq\langle \mathbb{Z} \rangle) = res : Bool { requiere \{ \text{true} \} asegura \{ res = \text{true} \leftrightarrow (\forall i,j:\mathbb{N}) \ (i < j \land j < |xs| \longrightarrow_L xs[i] \leq ys[j]) \} }
```

2. Testing

Los siguientes ejercicios tienen ya un código que los implementa correctamente. Su objetivo es generar un conjunto de tests que cubra todos los casos de uso.

Ejercicio 6. Crear tests para la función fizzBuzz, que dado un número natural devuelve un String según corresponda:

- Si es múltiplo de 3 y 5: "FizzBuzz"
- Si es múltiplo de 3 pero no de 5: "Fizz"
- Si es múltiplo de 5 pero no de 3: "Buzz"
- Si no es múltiplo ni de 3 y ni de 5: devuelve el número como String

Ejercicio 7. Crear tests para la función numeroCombinatorio.

```
proc numeroCombinatorio (in n: \mathbb{N}, in k \mathbb{N}) = res : \mathbb{N} { requiere \{\text{true}\} asegura \{res=\binom{n}{k}\}
```

Ejercicio 8. Crear tests para la función repeticionesConsecutivas, que devuelve el tamaño de la subsecuencia más larga de un arreglo tal que todos los elementos de la subsecuencia son iguales.

```
\begin{array}{l} \operatorname{proc\ repeticionesConsecutivas\ (in\ xs:\ seq\langle\mathbb{Z}\rangle) = \operatorname{res}:\mathbb{N}\ \left\{ \\ \operatorname{requiere}\ \left\{\operatorname{true}\right\} \\ \operatorname{asegura}\ \left\{\operatorname{existeSecuencia}(\operatorname{res},xs) \land \neg \operatorname{existeSecuencia}(\operatorname{res}+1,xs)\right\} \\ \operatorname{pred\ existeSecuencia\ (largo:\mathbb{N},\ xs:\ seq\langle\mathbb{Z}\rangle)}\ \left\{ \\ \left(\exists\ \operatorname{posInicial}:\mathbb{N}\right)\left( \\ \operatorname{posInicial}+\operatorname{largo} \leq |xs| \land_L\left(\forall i:\mathbb{N}\right)\left( \\ i < \operatorname{largo} \longrightarrow_L xs[\operatorname{posInicial}] = xs[\operatorname{posInicial}+i] \\ \right) \\ \right) \\ \left\} \\ \right\} \end{array}
```

3. Manejo de archivos

Ejercicio 9. Crear una función leerVector, que dado un Scanner y un largo: IN, lee un arreglo de tamaño largo del scanner.

Ejercicio 10. Crear una función leerMatriz, que dado un *Scanner* y *filas, columnas* : N, lee una matriz de tamaño *filas* × *columnas* del scanner. Pensar como reutilizar el código del ejercicio anterior.

Ejercicio 11. Crear una función imprimirPiramide, que dado un *PrintStream* y *alto*: N, imprime una pirámide con el alto pedido en el stream. Una pirámide es un texto con espacios y asteriscos, con *alto* cantidad de filas, donde cada fila tiene una cantidad impar creciente de asteriscos centrados por espacios. Por ejemplo, la pirámide de *alto* = 4 es:

```
***
****
****
```

Notar que la primer línea tiene 3 espacios a la izquierda, 1 asterisco y 3 espacios más a la derecha.