Introducción a la Programación

Guía Práctica 6
Testing de caja Negra

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Ejercicio 1. Pensar y responder las siguientes preguntas:

1. ¿Cuál es el objetivo de realizar testing de un software?
2. ¿Realizar testing sobre un software nos demuestra que el software funciona correctamente? Justificar.
3. ¿En qué consiste el testing de caja negra?. ¿Cuál es su principal característica?
4. ¿Se puede realizar testing de caja negra sin contar con una especificación del programa a testear? Justificar.

Ejercicio 2. Pensando en el test de caja negra utilizando el método de partición por categorías de un programa que, dados tres enteros que se interpretan como la longitud de cada uno de los lados de un triángulo, dice si el triángulo resultante es isósceles, escaleno o equilátero. Tener en cuenta que para que un triangulo sea factible, la suma de dos de sus lados debe ser mayor a la longitud del tercer lado.

1. Indicar cuáles son los Factores del programa.
2. ¿Existen Factores que son relaciones entre otros Factores? ¿Cuáles?
3. Pensar características relevantes (Categorías) para cada uno de los factores encontrados.
4. Determinar Elecciones (Choices) para cada Categoría de cada Factor.
5. Clasificar las Elecciones: errores, únicos, restricciones, etc
6. Armar los casos de prueba, combinando las distintas Elecciones determinadas para cada Categoría detallando el resultado esperado en cada caso.

Ejercicio 3. Dada la siguiente especificación del problema parteEntera (Ej 2 de la Guía 4):

`problema parteEntera` (x: $\mathbb{R}$) : $\mathbb{Z}$ {
  `requiere:` { $True$ }
  `asegura:` { $resultado \leq x < resultado + 1$ }
}

1. Diseñar los casos de test de caja negra utilizando el método de partición por categorías
2. Proveer datos de test para cada uno de los casos presentados.

Ejercicio 4. Dada la siguiente especificación del problema quitarTodos (Ej 2.6 de la Guía 5):

`problema quitarTodos` (e: T, s: $seq\langle T \rangle$) : $seq\langle T \rangle$ {
  `requiere:` { $True$ }
  `asegura:` { *resultado* es igual a s pero sin el elemento e. }
}

1. Diseñar los casos de test de caja negra utilizando el método de partición por categorías
2. Proveer datos de test para cada uno de los casos presentados.

Ejercicio 5. Dada la siguiente especificación del problema sumarN (Ej 3.4 de la Guía 5):

`problema sumarN` (n: $\mathbb{Z}$, s: $seq\langle \mathbb{Z} \rangle$) : $seq\langle \mathbb{Z} \rangle$ {
  `requiere:` { $True$ }
  `asegura:` {$|resultado| = |s| \land (\forall i : \mathbb{Z})(0 \leq i < |s| \rightarrow resultado[i] = s[i] + n$ }
}

1. Diseñar los casos de test de caja negra utilizando el método de partición por categorías

2. Proveer datos de test para cada uno de los casos presentados.

Ejercicio 6. Diseñar los casos de test de caja negra utilizando el método de partición por categorías para los siguientes problemas:

1. `multiplosDeN :: Integer -> [Integer] -> [Integer]` que dado un número n y una lista xs, devuelve una lista con los elementos de xs múltiplos de n. (Ej 3.8 de la Guía 5)

2. `ordenar :: [Integer] -> [Integer]` que ordena los elementos de la lista en forma creciente. (Ej 3.9 de la Guía 5)

3. `aplanarConNBlancos :: [[Char]] -> Integer -> [Char]`, que a partir de una lista de palabras y un entero n, arma una lista de caracteres concatenándolas e insertando n blancos entre cada palabra (n debe ser no negativo). (Ej 4.7 de la Guía 5)