Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información.
CI–3661 – Laboratorio de Lenguajes de Programación.
Septiembre–Diciembre 2012

Problemario de Semana I: Introducción y Funciones

1. Para las siguientes funciones escritas en el lenguaje de programación C, diga si las mismas con *puras* o *impuras*

   a) La función `fact` en:

   ```
   int fact(int n) {
     if (n > 0)
       return n * fact(n - 1);
     return 1;
   }
   ```

   b) La función `increment` en:

   ```
   int k;

   int increment() {
     return ++k;
   }
   ```

   c) La función `add_n` en:

   ```
   int k;

   int add_n(int n) {
     return k + n;
   }
   ```

   d) La función `set_char` en:

   ```
   char get_char(char *a, int i, char val) {
     char ret = a[i];
     a[i] = val;
     return ret;
   }
   ```

   e) La función `foo` en:

   ```
   int foo(int *a, int *b) {
     return bar(a) + bar(b);
   }

   int bar(int *a) {
     return a[0]++;
   }
   ```

2. Para cada una de las siguientes funciones, diga que conjuntos representan su dominio y su rango.

   Puede suponer que U es el conjunto que contiene a todos los conjuntos.

   a) Dado un número real, calcular su parte entera.

   b) Dado dos números enteros, decidir si el primero es mayor que el segundo.

   c) Dados dos conjuntos, calcular su intersección.

   d) Dada una función de enteros en enteros, calcular la cantidad de elementos en su dominio tal que la función evalúe a cero.

   e) Dado un conjunto A y un conjunto B, devolver el conjunto de todas las funciones que tienen dominio A y rango B.

   f) Dado un conjunto, devolver todos los posibles subconjuntos del mismo.

   g) Dada una función, devolver su función inversa (si existe).

3. Implemente, en el lenguaje de programación *imperativo* de su elección, las siguientes subrutinas usando solamente iteración y solamente recursión.

   a) Sumar los elementos de un arreglo.

   b) Calcular el n–ésimo número de Fibonacci.

   c) Ordenar un arreglo de n elementos, usando *Selection–Sort*.

4. Ingrese las siguientes expresiones a GHCI y observe los resultados.

   - `3 + 4`
   - `sum [1..10]`
   - `take 20 [1,3..]`
   - `(/2) 9`
   - `(2/) 9`
   - `product [1..5]`
   - `filter even [1..20]`
   - `takeWhile (<100) [x*x | x <- [1..]]`
   - `zip [1,2,3] ['a', 'b', 'c', 'd']`
   - `[(x, y) | x <- [1,2,3], y <- ['a', 'b', 'c', 'd']]`
   - `let f = 1 : scanl (+) 1 f in take 10 f`
   - `putStrLn "Hello World!"`