Universidad Nacional de Costa Rica (UNA) Escuela de Informática
EIF400 Paradigmas de Programación,
II Semestre 2022.
Dr. Carlos Loría-Sáenz.

Práctica Para I Parcial Versión (basada en exámenes semestres anteriores)

Advertencia: No asuma que el examen definitivo será necesariamente idéntico a esta práctica.

Nota: Cada pregunta viene acompañada de un anexo en la carpeta anexos adjunta según la pregunta lo requiere. Incluye ya el esqueleto de lo pedido para que Ud. lo complete.

1) a) Escriba en `Java` un método `<T> Predicate<T> implies(Predicate<T> p, Predicate<T> q)` que retorne un predicado que se comporta como $p \rightarrow q$. (es decir, como p implica q). Hágalo sin usar ninguna "arrow" (`->`). Ponga su respuesta en un script `predicates.jsh`.

   b) Lo mismo anterior buscando algo lo más parecido en ES6

2) Escriba una `class IterTools` en `ES6` con un método estático `*iterate(seed, f)` que de manera "lazy" genere la secuencia infinita $s_0 = seed; s_n = f(s_{n-1})$ para $n > 0$. Póngala en un módulo `itertools.mjs` (estilo `ESM`) que la exporte. Puede usar programación imperativa sólo donde es indispensable.

3) Haga lo mismo anterior en `Java`. Su método debe tener el tipo `<T> Iterator<T> iterate(T seed, UnaryOperator<T> f)`. Ponga su respuesta en un script del `Jshell` de nombre `itertools.jsh`. Ver ejemplo de uso en anexo.

4) Escriba en `JS` una función `cousins(obj1, obj2)` que retorne `true` si `obj1` y `obj2` son objetos primos, es decir, si sus padres (constructores) en la cadena prototípica son hermanos (es decir, tienen exactamente el mismo padre). Retorna `false` en otro caso. Asuma que todos los constructores se crean con `class` y usando `extends` para herencia. Verifique que `obj1` y `obj2` no son nulos o indefinidos. Verifique que no son el mismo objeto. En ambos casos si una verificación falla se retorna `false`. Su solución no debe fallar tratando de pedir una propiedad a algo `undefined` o `null`. Se califica claridad del código. Ponga su respuesta en un módulo `oop.mjs` (estilo `ES6`).

5) Considere la siguiente secuencia recurrente, donde asumiremos que a es un número positivo. Asumiremos que nunca x_n es cero para todo n. Esa secuencia se aproxima (converge) a la $\sqrt{a}$ cuando n crece suficiente ($n \rightarrow \infty$). Para este ejercicio puede usar paradigma imperativo, pero sólo en donde sea inevitable. La secuencia es así:

$$x_0 = \frac{a}{2};\ x_n = (x_{n-1} + \frac{a}{x_{n-1}})/2 \text{ para todo } n > 0.$$

   a) Escriba un generador en `ES6` `seqRoot(a)` que genera de forma infinita objetos de la forma $\{n: k, x: x_k\}$, es decir, cada objeto tiene en la propiedad n el número k de la secuencia, y en la propiedad x lleva el k-ésimo término de la secuencia.
   b) Escriba `first(a, test)` que retorna el primer objeto $\{n, x\}$ que `seqRoot(a)` generaría que cumple con una lambda `test` cualquiera.

c) Escriba `root(a, {max=20, epsilon=1e-10})` que usando `first` aproxime la raíz cuadrada de `a`. La lambda `test` a usar en `first` verifica que se genera a lo más un objeto de la forma $\{n: max, x: x_{max}\}$ o un objeto $\{n: k, x: x_k\}$ que cumple que $\left|x_k^2 - a\right| < epsilon$.

Haga un módulo `ES6` llamado `seqroot.mjs` con las tres funciones pedidas.

6) Revise y use el script de `Jshell` de nombre `persons.jsh` (ver carpeta asociada con la pregunta) y revise el archivo `persons.json` entregado, que al leerse usando el método `readPersonsJSON()` carga el archivo en una lista de 1000 records tipo `Person`.
Usando sólo `FP` (sin usar `reduce` ni estado mutable) escriba un método:
`Optional<List<Person>> oldestWomen(List<Person> persons)`
donde el contenido del `Optional` de retorno (si lo hay) es la lista de los records de mujeres de mayor edad en el archivo json. Su solución debe ser $O(n)$ para considerarse correcta donde n es el largo de la lista de personas. Ponga su respuesta en un script de `Jshell` llamado `jsonquery.jsh`.