Taller de Programación con estado

Modelos y Paradigmas de Programación

Escuela de Ingeniería de Sistemas y Computación

Universidad del Valle

Profesor Juan Francisco DIAZ FRIAS*

A continuación Usted encontrará una serie de ejercicios que debe resolver **individualmente**. No se admite ninguna consulta con sus compañeros de otros grupos de trabajo, pero sí con el profesor. Esa es la hipótesis básica sobre la cual se fundamenta esta tarea. Cualquier violación a esta regla será considerada un intento de copia y causará la anulación del taller.

La solución a todos y cada uno de los puntos del taller se debe almacenar en un archivo `.oz` tal como se describe en cada punto. Usted debe entregar como solución al examen, el conjunto de esos archivos empaquetado y comprimido en un solo archivo cuyo nombre debe ser análogo a `taller3JFDiaz.zip` o `taller3JFDiaz.tgz` o `taller3JFDiaz.tar.gz` o `taller3JFDiaz.rar`. Al desempaquetar este archivo, debo encontrar solamente los archivos siguientes: `sum.oz`, `fact.oz`, `fibo.oz`, `bubble.oz`, `diccionario.oz`, `arreglo.oz` y `amigos.oz`, con el contenido descrito abajo. Adicionalmente debe entregar un archivo `practica3.pdf`, con el desarrollo del taller que no corresponde a código Oz.

No envíe ningún archivo adicional, así lo haya utilizado para sus pruebas. Su solución debe ser enviada a través del Campus Virtual en el enlace correspondiente al Taller de Programación Concurrente Declarativa según las fechas indicadas allí. El sistema no permitirá que se envíen soluciones después de dicha fecha.

*juanfco.diaz@correounivalle.edu.co

1. Considere el siguiente fragmento de código:

```
declare
A={NewCell 0}
B={NewCell 0}
T1=@A
T2=@B
{Show A==B} % a).
{Show T1==T2} % b).
{Show T1=T2} % c).
A:=@B
{Show A==B} % d).
```

   Indique que valor será imprimido en cada una de las marcas *a), b), c) y d)* **true**, **false**, `A`, `B` ó 0. Explique claramente.

2. La función **fun** `{Q A B} ...` **end** calcula iterativamente la suma de los elementos entre `A` y `B` (`A` y `B` incluidos). Implemente `Q` de tal manera que use celdas para calcular la suma.

   Su solución debe guardarla en un archivo de nombre `sum.oz`.

3. Implemente la función que calcula el factorial de un número dado mediante celdas.

   Su solución debe guardarla en un archivo de nombre `fact.oz`.

4. La función de Fibonacci está definida de la siguiente manera:

$$\begin{aligned} fib(n) &= 1 \qquad \text{Si } n < 2 \\ fib(n) &= fib(n-1) + fib(n-2) \qquad \text{Si } n \geq 2 \end{aligned}$$

   Implemente esta función utilizando celdas.

   Su solución debe guardarla en un archivo de nombre `fibo.oz`.

5. Implemente el algoritmo de ordenamiento *BubbleSort* utilizando celdas.

   Su solución debe guardarla en un archivo de nombre `bubble.oz`.

6. *Amigos y dinero.* Nuestra triste historia comienza con un apretado grupo de amigos. Juntos realizaron un viaje al pintoresco país de Molvania. Durante su estancia, ocurrieron diversos acontecimientos que son demasiado horribles para mencionar. El resultado final fue que la última noche el viaje terminó con un cambio trascendental de "Yo nunca quiero verte de nuevo! "s. Un cálculo rápido nos dice que esta frase fue dicha casi 50 millones de veces!

   De regreso en Escandinavia, nuestro grupo de ex-amigos se dan cuenta de que no se han dividido los gastos realizados durante el viaje de manera uniforme. Algunas personas pueden deber varios miles de coronas. La liquidación de las deudas tiende a ser un poco más problemática de lo que debería ser, ya que muchos en el grupo ya no desean hablarse, y menos aún a dar dinero entre sí.

Para dar solución a este problema, se le pide a cada persona que diga cuánto dinero debe o le es debido y de que personas sigue siendo amigo. Teniendo en cuenta esta información, es pobile determinar si todo el mundo puede pagar su deuda o recibir el dinero que le es debido pasando dinero solamente entre personas que aún siguen siendo amigos.

Implemente la función `{PagoDeuda Amistades Deudas}`. Donde `Amistades` corresponde a una lista de parejas `I#J` que representan que la persona `I` mantiene una relación de amistad con la persona `J` y `Deudas` corresponde a una tupla de valores enteros (positivos y negativos) donde la posición `I` en esta tupla corresponde al dinero que debe o le es debido a la persona `I` (positivo si debe dinero y negativo si le es debido). Esta función debe determinar si es posible que todas las deudas sean pagadas entre las personas si solo se pasa dinero entre personas que aún son amigas.

Puede asumir que la sumatoria de los valores en la tupla `Deudas` es igual a 0 y que el tamaño de esta tupla corresponde al número total de personas. Utilice las funciones auxiliares que considere necesarias.

Ej. Los siguientes llamados a la función `PagoDeuda`:

```
{Browse {PagoDeuda [1#2 2#3 4#5] [100 ~75 ~25 ~42 42]}}
{Browse {PagoDeuda [1#3 2#4] [15  20 ~10 ~25]}}
```

Deberán despleglar **`true`** y **`false`** respectivamente.

Su solución debe guardarla en un archivo de nombre `amigos.oz`.