Apellido y Nombre:
e-mail:

escrito	lab	nota

1	2	3	4	L

Algoritmos y Estructuras de Datos II **Examen Final** **30/7/2008**

Importante: Escribir nombre y apellido en todas las hojas, inclusive ésta que también debe ser entregada. Resolver cada ejercicio en hoja aparte.

1. (3pts) Especificar el TAD FileSystem de modo de que represente el estado de un sistema de carpetas. Tendrá tres constructores: uno (root) que crea el file system inicial consistente de una carpeta principal vacía (carpeta raíz), otro (mkdir) que agrega una carpeta a la carpeta actual y automáticamente esa nueva carpeta pasa a ser la carpeta actual y finalmente uno (up) que permite cambiar a la carpeta padre de la carpeta actual.

   TAD FileSystem
   constructores
   root :: FileSystem
   mkdir :: Nombre × FileSystem → FileSystem
   up :: FileSystem → FileSystem

   Por ejemplo,

   $$\left[\begin{array}{l} c1 \left[\begin{array}{l} c11 \left[\begin{array}{l} c111 \\ c112 \end{array}\right. \\ \underline{c12} \end{array}\right. \end{array}\right.$$

   donde c12 fue subrayada para indicar que es la carpeta actual, puede representarse por fs = mkdir(c12,up(up(mkdir(c112,up(mkdir(c111,mkdir(c11,mkdir(c1,root))))))))

   Leyendo de derecha a izquierda, primero se crea la raíz, luego c1 (y se cambia a c1), se crea c11 (y se cambia a c11), se crea c111 (y se cambia a c111), se sube (a c11), se crea c112 (y se cambia a c112), se sube (a c11), se sube (a c1), se crea c12 (y se cambia a c12).

   Otro ejemplo,

   $$\left[\begin{array}{l} \\ c1 \left[\begin{array}{l} c11 \left[\begin{array}{l} c111 \\ c112 \end{array}\right. \\ c12 \left[c123\right. \\ c13 \end{array}\right. \\ \underline{c2} \left[c21\right. \end{array}\right.$$

   puede representarse por up(mkdir(c21,mkdir(c2,up(up(mkfile(c13,up(up(mkdir(c123,fs))))))))))

   Si el constructor mkdir se aplica a un nombre que ya existe en la carpeta actual, su efecto es sólo cambiar a la carpeta en cuestión.

   a) Dar una operación path que se aplique a un FileSystem y devuelva el path de la carpeta actual, como una lista, por ejemplo, al aplicarse a fs devuelve c12 ▷ c1 ▷[].

   b) Dar una operación esraíz que se aplique a un FileSystem y determine si la carpeta actual es o no la carpeta principal (la raíz).

   c) Dar una ecuación entre constructores que exprese que si la carpeta actual es la carpeta raíz, entonces el constructor up no tiene efecto alguno.

   d) Dar una operacion existe que se aplique a un FileSystem y a un nombre y determine si existe una carpeta con ese nombre en la carpeta actual. Para ello puede convenir definir otra operación más general (difícil).

2. (2,5pts) Resolver el problema de la federación agraria: un productor tiene una cantidad de dinero D y quiere utilizarlo para almarcenar la mayor cantidad de soja posible. Hay n silos con capacidad para $t_1, \ldots, t_n$ toneladas y el alquiler de cada uno de ellos por el tiempo estimado cuesta $p_1, \ldots, p_n$. Dar un algoritmo que determina la mayor cantidad de soja almacenable sin exceder el dinero disponible D. Justificar. ¿Qué técnica utilizó?
3. (2pts) Se debe calcular el número de veces que el procedimiento p escribe la palabra "hola" en función de la entrada n. Para ello
   a) Definir la recurrencia correspondiente.
   b) Resolverla.

   Obs: recuerde que usted mismo puede comprobar que la recurrencia esté correctamente resuelta. El argumento de p puede ser cualquier número natural $n \geq 0$.

```
proc p(in n:nat)
    if n = 1 → for i:= 1 to 2 do escribir("hola") od
      n ≥ 2 → for i:= 1 to 2 do
                  p(n-1)
                  p(n-2)
              od
    fi
end
```

4. (2,5pts) Se tiene una grilla de 3 por 3 donde deben disponerse exactamente una vez cada dígito entre 1 y 9 de modo de que los dígitos en cada fila, cada columna y cada una de las 2 diagonales principales sume 15. Por ejemplo, la grilla

2	7	6
9	5	1
4	3	8

satisface dicha condición. El siguiente algoritmo cuenta el número de soluciones diferentes a este problema

```
fun nro_sols() ret n: nat
                          {calcula el número de grillas que satisfacen la condición}
  var grilla:array[1..3,1..3] of {1,2,3,4,5,6,7,8,9}
  n:= backtrack(grilla,{1,2,3,4,5,6,7,8,9},0,3)
end
fun backtrack(grilla,D, x : {1,2,3}, y : {1,2,3}) ret n: nat
            {x es la fila, y es la columna, D son los dígitos que quedan por asignar}
  n:= 0;
  if x = 3 ∧ y = 3 → if condición(grilla) then n:= 1 fi
    x < 3 ∨ y < 3 → if y = 3 → x:= x+ 1
                               y:= 1
                    y < 3 → y:= y+1
                    fi
                    for d ∈ D do
                      grilla[x,y]:= d
                      n:= n + backtrack(grilla, D - {d}, x, y)
                    od
  fi
end
```

Se pide explicar el algoritmo lo más detalladamente posible. Justificar claramente cada asignación a n y los valores de cada uno de los parámetros en cada llamada a la función backtrack.

Para alumnos libres: definir la función condición, que debe determinar si la condición de que la grilla sea una solución se cumple o no.