Algoritmos y Estructuras de Datos II - 29 de Abril de 2024
Primer Parcial

Alumno: .....

Siempre se debe explicar la solución, una respuesta correcta no es suficiente si no viene acompañada de una justificación que demuestre que la misma ha sido comprendida. Las explicaciones deben ser completas. En ejercicios de implementación, debe utilizarse el lenguaje de la materia. **La utilización de código c influirá negativamente.**
Por favor, desarrollá cada solución en **hojas diferentes** y escribí claramente **tu nombre** en todas las hojas, ya que las mismas se separarán para agilizar la corrección.

1. Se requiere desarrollar el software de un GPS para mapas de ciudades conformadas por manzanas cuadradas de 100 metros de lado y cuyas calles son todas de doble mano. El GPS dará indicaciones al finalizar cada cuadra, con las siguientes posibilidades: **continúe recto, gire a la izquierda, gire a la derecha, llegó al destino**.

   Para ello se debe definir un TAD Recorrido que contendrá las indicaciones del GPS para ir de una esquina de la ciudad hasta otra. Un recorrido trivial será el que consta de ir desde una esquina hasta ella misma, por lo cual simplemente el GPS dirá **llegó al destino**. Un ejemplo un poco más interesante podría ser: **continúe recto, continúe recto, gire a la izquierda, gire a la derecha, llegó al destino**. Un Recorrido entonces consistirá de una secuencia de indicaciones.

   Se necesita también que una vez realizada una acción el recorrido restante se actualice. Para ello, se introducen las operaciones **actualizar continuando recto**, **actualizar girando a la izquierda** y **actualizar girando a la derecha**, que en caso de coincidir con lo indicado por el recorrido se limitará a eliminar del mismo la indicación ejecutada, pero en caso de no coincidir deberá recalcular el recorrido para llegar al destino original.

   Por ejemplo, si en el último ejemplo el vehículo continuó recto, el recorrido resultante será: **continúe recto, gire a la izquierda, gire a la derecha, llegó al destino**. En cambio, si el vehículo en vez de continuar recto, giró a la izquierda, el recorrido resultante podrá ser entonces: **gire a la derecha, gire a la derecha, gire a la izquierda, gire a la izquierda, gire a la derecha, llegó al destino**.

   También se requieren operaciones que devuelvan un valor booleano para saber si la siguiente indicación de un recorrido es continuar recto, doblar a la derecha, doblar a la izquierda o si ya se llegó al destino.

   (a) Especificá el TAD Recorrido con los constructores correspondientes, las tres operaciones de actualización, las cuatro operaciones de consulta, y una operación **longitud**, que devuelve la longitud medida en metros de un recorrido.

   (b) Implementá el TAD Recorrido utilizando una lista de elementos de tipo **Indicación**, donde el tipo Indicación deberá definirse como un enumerado con 3 valores: Straight, Left, Right.

   (c) Utilizando el tipo de datos abstracto Recorrido, implementá una función que dado un Recorrido, devuelva un natural indicando cuántas veces se debe doblar a la izquierda.

2. Utilizando la siguiente implementación del tipo *list* con punteros, se quiere implementar el procedimiento *lshift*. Este procedimiento toma una lista y efectua una rotación circular a izquierda de los elementos. Por ejemplo, dada la lista de números $[0, 0, 1, 1, 0, 1]$, al aplicar el procedimiento *lshift*, el resultado será $[1, 0, 0, 1, 1, 0]$.

```
type node = tuple
                value: elem
                next: pointer to node
            end
type list = pointer to node
```

   Implementá el procedimiento *lshift* dando correctamente su encabezado y definición.

3. Para cada uno de los siguientes algoritmos determinar **por separado** cada uno de los siguientes incisos.

   (a) ¿Qué hace? ¿Cuáles son las precondiciones necesarias para que haga eso?

   (b) ¿Cómo lo hace?

   (c) El orden del algoritmo, analizando los distintos casos posibles.

   (d) Proponer nombres más adecuados para los identificadores (de variables, funciones y procedimientos).

```
proc p (in/out a: array[1..N] of T)
    var i: nat
    var x: nat
    i:= 2
    do i <= N
        x:= f(a,i)
        swap(a,i,x)
        i:= i+2
    od
end proc
```

```
fun f (a: array[1..N] of T, i: nat) ret x: nat
    var j: nat
    j:= i+2
    x:= i
    do j <= N
        if a[j] < a[x] then x:= j fi
        j:= j+2
    od
end fun
```

4. (a) Dado el siguiente arreglo, muestre cómo queda el mismo luego de cada modificación que realiza el algoritmo **quick sort**.

$$[7, 20, 4, 6, 2, 15, 3]$$

   (b) Da las secuencias de llamadas al procedimiento *quick_sort_rec* indicando correctamente sus argumentos.