Algoritmos y Estructuras de Datos II – 8 de Febrero de 2021

Examen Final Teórico-Práctico

Alumno: ..... Email: .....

Siempre se debe explicar la solución. Una respuesta correcta no es suficiente si no viene acompañada de una justificación lo más clara y completa posible. Los algoritmos no deben escribirse utilizando código c o de bajo nivel, sino el código de la materia y evitando la utilización innecesaria de punteros. La no observación de estas recomendaciones resta puntaje.

1. (Voraz) Anacleta reparte pedidos en bicicleta. Todos los clientes viven en la Avenida San Wachín y la dirección del trabajo de Anacleta es Avenida San Wachín 0. Al llegar a trabajar, Anacleta tiene cada pedido con la altura de la casa donde debe entregarlo. En la bici, Anacleta puede transportar cinco pedidos o menos. Escriba un algoritmo que reciba una lista de naturales llamada pedidos y determine la menor distancia que debrá pedalear Anacleta para entregar todos los pedidos.

2. (Backtracking) Se tienen n objetos de peso $p_1, \ldots, p_n$ respectivamente. Se tiene una mochila de capacidad K. Dar un algoritmo que utilice backtracking para calcular el menor desperdicio posible de la capacidad de la mochila, es decir, aquél que se obtiene ocupando la mayor porción posible de la capacidad, sin excederla. Definir primero en palabras la función aclarando el rol de los parámetros o argumentos.

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 ello?

   (b) ¿Cómo lo hace?

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

   (d) Proponer nombres más adecuados para las funciones.

```
fun s(v: nat, p: array[1..n] of nat) ret y: nat
    y:= v
    while y < n ∧ p[y] ≤ p[y+1] do
        y:= y+1
    od
end fun
```

```
fun t(p: array[1..n] of nat) ret y: nat
    var z: nat
    y, z:= 0, 1
    while z ≤ n do
        y, z:= y+1, s(z,p)+1
    od
end fun
```

```
fun u(p: array[1..n] of nat) ret v: bool
    v:= (t(p) ≤ 1)
end fun
```

4. Completar la especificación del tipo Conjunto de elementos de tipo T, agregando las operaciones

   - **elim**, que elimina de un conjunto un elemento dado.
   - **union**, que agrega a un conjunto todos los elementos que pertenecen a otro conjunto dado.
   - **dif**, que elimina de un conjunto todos los elementos que pertenecen a otro conjunto dado.

```
spec Set of T where

constructors
    fun empty_set() ret s : Set of T
    {- crea una pila vacía. -}

    proc add (in e : T, in/out s : Set of T)
    {- agrega el elemento e al conjunto s. -}

destroy
    proc destroy (in/out s : Set of T)
    {- Libera memoria en caso que sea necesario. -}

operations
    fun is_empty_set(s : Set of T) ret b : bool
    {- Devuelve True si s es vacío. -}
```

fun cardinal(s : Set **of** T) **ret** n : nat
{- *Devuelve la cantidad de elementos que tiene* s -}

fun member(e : T, s : Set **of** T) **ret** b : bool
{- *Devuelve True si el elemento* e *pertenece al conjunto* s -}

proc inters (**in**/**out** s : Set **of** T,**in** s0 : Set **of** T)
{- *Elimina de* s *todos los elementos que NO pertenecen a* s0 -}

{- **PRE:** not is_empty_set(s) -}
fun get(s : Set **of** T) **ret** e : T
{- *Devuelve algún elemento (cualquiera) de* s -}

5. A partir de la siguiente implementación de conjuntos utilizando listas ordenadas, implemente el constructor **add**, y las operaciones **get**, **inters** y **elim**. La implementación debe mantener el invariante de representación por el cual todo conjunto está representado por una lista ordenada crecientemente. Puede utilizar todas las operaciones especificadas para el tipo lista vistas en el teórico. Para cada operación que utilice, especifique su encabezado, es decir: si es función o procedimiento, cómo se llama, qué argumentos toma y devuelve.

   implement Set **of** T **where**

   type Set **of** T = List **of** T

   fun empty_set() **ret** s : Set **of** T
       s := empty_list()
   end fun

6. (Para alumnos libres) ¿Qué hace el algoritmo de Prim? ¿Qué hace el algoritmo de Dijkstra?