Universidad Rey Juan Carlos

Estructuras de Datos
Mayo 2019

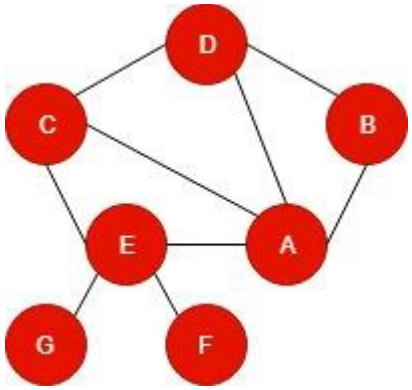
APELLIDOS: ***NOMBRE:***
TITULACIÓN: ***__GII __GII+ADE __GII+CRI __GII+MAT __GIS __GIS+GII __GIS+MAT***
DNI:
Duración: 1:45 h.
Calificación:

Se recuerda que, según la guía docente, el 25% se corresponde con prueba escrita teórica y 60% con prueba escrita práctica. El 15% restante proviene de las pruebas prácticas realizadas a lo largo del curso. Se sugiere dejar 1 hora de esfuerzo en la resolución del ejercicio 5. Contestar en la hoja de enunciados TODOS los ejercicios.

1) (**Prueba escrita teoría, 1.5 punto**) Represente gráficamente el árbol AVL cuyos recorridos preorden y postorden son [20, 15, 10, 5, 17, 19, 25, 30] y [5, 10, 19, 17, 15, 30, 25, 20] respectivamente. A continuación, describir el proceso de inserción del nodo con clave 27, mostrando todos los pasos intermedios.

2) (**Prueba escrita práctica, 0.5 puntos**) Dada la implementación estática de `TBolsa`, se pide implementar la operación Diferencia que dadas dos `TBolsas`, devuelve en la primera `TBolsa` la Diferencia entre las dos `TBolsas`. Si la primera bolsa tiene 3 bolas blancas y la segunda 2, la diferencia dejara 1 única bola blanca en la primera bolsa. Se consideran que las dos bolsas tienen tamaño N. La interfaz de la operación Diferencia es: `PROCEDURE Diferencia(VAR b1 : TBolsa; b2 : TBolsa);`

3) (**Prueba escrita teoría, 1 punto**). Dado el siguiente grafo, se pide:



Recorrer en profundidad el grafo, aplicando el algoritmo visto en clase, empezando por el nodo A. Es necesario mostrar el estado, paso a paso, de las estructuras de datos auxiliares utilizadas. Los adyacentes a un nodo se devuelven en orden alfabético.

4) (**Prueba escrita práctica, 1,5 puntos**) Sobre árboles binarios:

Implemente en una unidad la definición de tipos de un árbol binario en memoria dinámica que guarde elementos de tipo `TElemento` definido en la unidad `uElemento`. A continuación implemente las operaciones necesarias para implementar el procedimiento `TioSinHijos`. Este procedimiento nos devuelve una lista con todos los nodos que sean tios de un nodo, pero que a su vez no tengan hijos. Un nodo es tio si su hermano tiene hijos.

5) (Prueba escrita práctica, 4 puntos)

Con motivo de la celebración de elecciones, la empresa pública *Mails* quiere digitalizar el funcionamiento en sus oficinas ya que, año tras año, el número de oficinas crece y esto dificulta su gestión. En cada oficina, solo nos interesa el funcionamiento de cómo se atiende a los clientes. Cada oficina tiene tres tipos de servicio: envío, recepción y votar por correo, y cada uno de estos tipos de servicio se atiende por estricto orden de llegada del cliente.

Por otro lado, al cliente se le pide que lleve el DNI en la mano, para evitar demoras, y que saque ticket en función del servicio que ha solicitado, el cuál asignará al cliente a la ventanilla menos saturada. Cada ventanilla puede tener clientes con diferentes servicios.

a) **[0,5 puntos]** Se pide diseñar las estructuras de datos necesarias para poder gestionar las oficinas de Mails. El tipo de dato principal se llamará `TMails` y contendrá un listado con las `TOficinas` que hay en el territorio nacional. De cada `TOficina` nos interesa saber: su identificador, su código postal y el número de trabajadores. Además, en cada `TOficina` se dispone de una serie de ventanillas que atienden a los clientes por orden de llegada. Cada `TOficina` tendrá 5 `TVentanillas`. Cada `TVentanilla` tendrá un responsable (string), un estado (abierto o cerrado) y un código único para todas las oficinas (integer). De los `TCliente` que llegan a las ventanillas, nos interesa saber su DNI, y el servicio que ha solicitado.

b) **[1 puntos]** Crear los procedimientos adecuados para añadir un `TCliente` a la `TVentanilla` abierta, menos saturada. Para ello se dispone del DNI del cliente, el tipo de servicio que desea recibir y el identificador de la oficina en la que está solicitando el servicio. Para determinar la ventanilla idonea, se puede usar la función LongitudVentanilla, que devuelve el número de clientes en esperan en dicha ventanilla, su prototipo es:

```
FUNCTION LongitudVentanilla(cod: integer) : integer;
{PRE: "cod" es el código único de la TVentanilla. La función devuelve
el número de clientes en espera.}
```

c) **[1 puntos]** En determinadas circunstancias, es necesario cerrar una `TVentanilla`, y dado que es posible que hubiera algún `TCliente` en ella, es necesario repartir los `TClientes` en espera, entre el resto de `TVentanillas` que estén abiertas. Implementar todos los procedimientos necesarios para realizar la operación anterior, sabiendo que se nos pasa el código único de la `TVentanilla` a cerrar. Nota: los `TClientes` son asignados por orden, se extrae el cliente que lleva más tiempo en la `TVentanilla` a cerrar y se le asigna a la `TVentanilla` abierta y menos saturada como último cliente. Se repite la operación hasta que la `TVentanilla` a cerrar esté vacía.

d) **[1,5 puntos]** En época de elecciones, cuando llega la hora habitual de cierre de una oficina y para evitar saturaciones, se cierran todas las `TVentanillas` menos una. Los `TClientes` actualmente en espera cuyo servicio solicitado sea "votar", serán añadidos a la `TVentanilla` que permanecerá abierta, descartando al resto de `TClientes` que tendrán que volver otro día. Se sigue un estricto orden a la hora de asígnar los `TClientes` a la `TVentanilla` única. Primero se asignan los primeros `TClientes` de cada ventanilla: el primer `TCliente` de la primera `TVentanilla`, el primer `TCliente` de la segunda `TVentanilla`, el primer `TCliente` de la tercera `TVentanilla`, etc. Cuando se han reasignado los primeros `TClientes` de cada `TVentanilla` pasa a asignarse los segundos (que ahora serán los primeros) y así hasta que solo queden `TClientes` en la `TVentanilla` única. Se pide implementar los procedimientos adecuados para cerrar todas las ventanillas menos una, permitiendo a los `TCliente` cuyo servicio solicitado haya sido "votar", en el orden descrito anteriormente.