Práctico 4

Estructuras múltiples

Clasificación de ejercicios:
(I) Imprescindibles
(R) Recomendados
(C) Complementarios

EJERCICIO 1 (I)

Periódicamente la FIFA (máximo organismo mundial del fútbol) publica una clasificación de los países asociados. El primero en la clasificación es el país cuya selección es considerada la más poderosa del mundo, mientras que el último es aquel país cuya selección es considerada la más débil. Por ejemplo la clasificación para el mes de Setiembre de 2004 podría ser:

```
  1   Brasil
  2   Francia
  3   España
  …   …
 19   Uruguay
  …   …
```

Se desea implementar las siguientes funciones:

- **int queLugar(pais p, clasificacion L)**, en el cual dado el nombre de un país y una clasificación, devuelve el lugar en la clasificación en donde está dicho país, ó 0 si dicho país no está en la clasificación. Esta operación debe realizarse en O(1) en promedio.

- **pais quePais(int k, clasificacion L)**, en el cual dado un entero y una clasificación, devuelve el nombre del país que ocupa el lugar k en la clasificación, ó "" (string vacío) si k es mayor que la cantidad de países clasificados. Esta operación debe realizarse en O(1) en el peor caso.

- **clasificacion cambio(pais p, int k, clasificacion L)**, en el cual dado un país, un entero y una clasificación, mueve dicho país k posiciones respecto a su lugar original, y lo intercambia con el país que ocupa la nueva posición. Por ejemplo una llamada a cambio (Uruguay, -17, L), deja la clasificación del ejemplo como sigue:

```
  1   Brasil
  2   Uruguay
  …   …
 19   Francia
  …   …
```

Notar que el mismo resultado se hubiera obtenido con la llamada cambio(Francia, 17, L). Esta operación debe realizarse en O(1) en promedio.

- **clasificacion afiliacion(pais p, clasificacion L)**, en el cual dado el nombre de un país y una clasificación, ingresa el nuevo país en el último lugar de la clasificación. Se asume que el país a afiliar no está en la clasificación. Esta operación debe realizarse en O(1) en el peor caso.

- **clasificacion desafiliacion(pais p, clasificacion L)**, en el cual dado el nombre de un país y una clasificación, quita a dicho país de la clasificación y sube la posición de todos los países que estaban por debajo suyo un lugar más arriba en la clasificación.

Hay un máximo posible de N países, con N acotado pero indeterminado.

Se pide:
a. Diseñar estructuras de datos apropiadas para implementar eficientemente dichas operaciones, describiendo como se obtienen las cotas de tiempo pedidas.
b. Dar una declaración en C de los tipos de datos descritos en la parte a).
c. Escribir en C la función desafiliación.

EJERCICIO 2 (I)

Una compañía de seguros maneja información relacionada con sus afiliados y los automotores de los mismos, así como de los accidentes que sufren dichos automotores durante un determinado año. La información relativa a los propietarios consiste en su nombre y cédula de identidad. La cantidad de propietarios está acotada, y se identifican por su cédula de identidad. De cada auto se conoce marca, modelo, matrícula y el mes de último pago de su seguro. Los autos se identifican por su número de matrícula. Un propietario puede tener un máximo de 5 autos a su nombre. De los accidentes en un mes, no se conoce un número máximo, y se identifican por el número de matrícula del auto del denunciante y el día del accidente. Se lleva también una descripción de la causa del choque.

Se desea satisfacer los siguientes requerimientos donde n es la cantidad de propietarios:
1. Dada la cédula de un propietario, saber que autos posee un propietario. Este requerimiento debe realizarse en O(1) promedio.
2. Dada la matrícula de un auto, conocer el nombre de su dueño. Este requerimiento debe realizarse en O(1) promedio.
3. Dada la matrícula de un auto, devolver la lista de los accidentes que tuvo en el año, ordenada por fecha. Este requerimiento debe realizarse en O(1) promedio.
4. Dada la fecha actual, listar los propietarios que están atrasados en el pago (aquellos cuyo mes de útimo pago es menor que el mes actual) y en qué coche. Dicho listado debe estar ordenado por nombre de propietario. Este requerimiento debe realizarse en O(n) peor caso.
5. Dado un mes devolver la lista de matrículas de autos accidentados en dicho mes ordenada por día. Este requerimiento debe realizarse en O(1) peor caso.
6. Dar de alta los datos de un accidente en la estructura. Se puede asumir que los accidentes son ingresados en el mismo orden histórico en que suceden. Este requerimiento debe realizarse en O(1) promedio.
7. Dados el nombre y la cédula de un propietario y los datos relativos a un auto, agregar dicho auto a la lista de autos que posee dicho propietario. En caso de que el propietario no figure en la estructura hay que darlo de alta. Este requerimiento debe realizarse en O(log n) promedio.

Se pide:
a. Diseñar estructuras de datos apropiadas para implementar eficientemente dichas operaciones, describiendo como se obtienen las cotas de tiempo pedidas.
b. Implementar el procedimiento 6, con el cabezal **void AltaAccidente(int dia, int mes, int matricula, string causa, Seguros *estructura),** donde Seguros es el tipo de datos que representa la estructura diseñada.

EJERCICIO 3 (R)

Se quiere implementar un sistema de asignación de empleados públicos a comisiones receptoras de votos en los diferentes circuitos de un acto electoral.

Un empleado público puede ser identificado tanto por su nombre completo como por su credencial cívica y está asociado a un organismo público. Cada organismo público está identificado por un número único y tiene un nombre asociado. Hay un máximo de K organismos públicos y M empleados por cada organismo (con M posiblemente muy grande).

Un empleado público puede estar asignado a un cargo en alguna comisión receptora de votos, o puede no estarlo. En caso de estar asignado, recibe una citación donde entre otras cosas indica el número de circuito asignado, el cargo que ocupa (identificado por un número entre 1 y 9) y el local en donde debe presentarse el día del acto electoral.

Un circuito está identificado por un número de circuito, y tiene asociado un local, un rango de credenciales de votantes en dicho circuito (por ejemplo votantes con credenciales AAA30981 al AAA31781), y la lista de los 9 miembros seleccionados. Hay un número N fijo de circuitos habilitados. Se desea implementar las siguientes operaciones:

- **void citacion(credencial c, int n, int cargo, estructura *E)**, que dados la estructura E, un número de credencial c, un número de circuito n, y un número de cargo, asigna al empleado con credencial c al circuito n en el cargo cargo. Esta operación se debe realizar en O(1) promedio.

- **void circuito(int n, estructura E)**, que dados un número de circuito n y la estructura E, imprime los nombres, credenciales, organismos en que trabajan y cargo que ocupa de los miembros de la comisión receptora de votos del circuito n. Esta operación se debe realizar en O(1) peor caso.

- **pareja citado(string quien, estructura E)**, que dados el nombre quien de un empleado público y la estructura E averigua si el empleado está citado. En caso afirmativo, devuelve la pareja formada por el número de circuito asignado y el cargo que ocupa. En caso negativo, devuelve la pareja (0,0). Esta operación debe realizarse en O(1) promedio.

- **ListaEmpleados organismo(int k, estructura E)**, que dados el número k de un organismo y la estructura E, devuelve la lista de empleados asignados, junto con su credencial, número de circuito, ubicación del circuito y cargo que ocupa. Esta lista debe estar ordenada alfabéticamente por nombre de funcionario. Esta operación debe realizarse en O(M) peor caso, donde M es la cantidad de funcionarios del organismo k.

- **ListaFunc fueraDeCircuito(estructura E),** que dada una estructura E, devuelve una lista de funcionarios asignados a ocupar cargos en circuitos distintos al que le correspondería votar. Estos son funcionarios cuya credencial cívica está fuera del rango asociado al circuito al cual fueron asignados. Esta lista debe ser impresa ordenada por número de circuito. Esta operación debe realizarse en O(N) peor caso, donde N es la cantidad de circuitos habilitados.

Se pide:
a. Diseñar estructuras de datos apropiadas para implementar eficientemente dichas operaciones, describiendo cómo se obtienen las cotas pedidas, y justificando sus decisiones.
b. Especificar la declaración de tipos en C.
c. Implementar en C las operaciones citado y organismo.

EJERCICIO 4 (C)

Se desea mantener información sbore un campeonato de fútbol que involucra a jugadores, equipos y partidos. De los jugadores se conoce su nombre, fecha de nacimiento y equipo al cual pertenecen. Todos los jugadores deben pertenecer a un equipo, y se asume que sus nombres son únicos. De los equipos se conoce el nombre (el cual los identifica), fecha de fundación y

jugadores que le pertenecen. Cada equipo tiene un máximo de K jugadores, y se asume que el total de jugadores entre todos los equipos es N y que hay una cantidad E de equipos en el torneo.

En el campeonato todos los equipos juegan contra todos a dos ruedas turnándose los derechos de locatario, con lo cual hay un total n.(n-1) partidos en el campeonato. Los partidos pueden ser jugados o por jugar. De los partidos por jugar interesa saber los equipos y la fecha, hora y estadio en el cual van a jugar. De los partidos jugados interesa además el resultado, las alineaciones de los equipos y la lista de goles efectuados.

Se desea satisfacer los siguientes requerimientos:

- **void nuevoJugador(string nomjug, date fnac, string equipo, estructura *E)**, en el cual dados el nombre de un jugador, su fecha de nacimiento y el equipo en el cual va a jugar, agrega dicha información a la estructura que representa el campeonato. Este requerimiento se debe realizar en O(log N) peor caso.

- **ListaJugadores listaJugadores(estructura E)**, en el cual dada la estructura que representa el campeonato, lista todos los jugadores que participan en él. El listado debe estar en orden alfabético y debe indicar el nombre del jugador y el equipo al cual pertenecen. Este requerimiento se debe realizar en O(N) peor caso.

- **void resultado(string neq1, string neq2, estructura E)**, en el que dados dos equipos lista el resultado del partido entre los equipos neq1 y neq2, cuando neq1 jugó como locatario, o la fecha, hora y estadio en el que van a jugar en caso de que aún no se haya jugado. Este procedimiento se debe realizar en O(1) caso promedio.

- **void transferencia(string nomjug, string neq, estructura *E)**, en el que dados el nombre de un jugador, y el nombre de un nuevo equipo neq, da de baja de su equipo actual, y lo da de alta el equipo neq. Este requerimiento se debe realizar en O(log N) promedio.

- **int puntosDeLocal(string neq, estructura E)**, en el que dado el nombre de un equipo, devuelve la cantidad de puntos como local que haya obtenido hasta el momento. Se le asigna 3 puntos por cada partido ganado, 1 punto por cada partido empatado y 0 punto por cada partido perdido. Este requerimiento se debe realizar en O(N) peor caso.

Se pide:
a. Definir las estructras necesarias para resolver los requerimientos anteriores en las cotas de tiempo pedidas, indicando como se satisfacen dichas cotas.
b. Especificar las declaraciones de tipos.
c. Implementar el requerimiento puntosDeLocal en C.

EJERCICIO 5 (C)

Se quiere diseñar un sistema de asignación de pasajeros y vuelos a diferentes viajes de una compañía aérea en un año dado.

Los aeropuertos son identificados por un conjunto de 3 letras (ejemplo, YYZ). Cada vuelo dentro de una compañía está identificado por un número de vuelo de 3 dígitos. Un vuelo determinado, tiene asociado un aeropuerto de origen y uno de llegada, junto con sus respectivos horarios de salida y llegada (ejemplo, el vuelo 903 de los lunes sale a las 8:10 de YYZ y llega el mismo lunes a las 12:10 a MIA). Un mismo número de vuelo puede salir más de una vez en una semana (ejemplo, el vuelo 903 sale los lunes, miércoles y viernes a las 8:10 de YYZ y llega a las 12:10 del mismo dia a MIA). Notar que los vuelos están asociados a días de una semana, y no a fechas concretas. La compañía tiene un promedio de 700 vuelos diarios.

Para una fecha dada, un vuelo tiene además asociado un viaje (ejemplo, el viaje del vuelo 903 del día viernes 6 de marzo de 1998). Un viaje tiene además asociado una capacidad (cantidad máxima

de pasajeros que puede cargar), una lista de pasajeros que tienen lugar reservado, y una lista de espera en el caso de que el viaje esté lleno.

Los pasajeros son identificados unívocamente por su nombre. Un pasajero tiene reserva de uno o más viajes en la compañia. Puede tener lugar confirmado, o puede estar en una lista de espera en el caso que el viaje que desee esté lleno. Hay un máximo M de pasajeros.

Se desea satisfacer los siguientes requerimientos, donde el tipo estructura es el tipo de datos que define la estructura a diseñar en el problema:

- **void asignacion(date fecha, int vuelo, string pasajero, estructura *E)**, en que dada una fecha (día/mes), el número de vuelo y el nombre de un pasajero, le reserva un lugar al pasajero para un viaje en dicho vuelo a realizarse en la fecha dada. En caso de que el viaje esté lleno, lo agrega al final de la lista de espera. Se puede asumir que el vuelo existe en el día dado (o sea que si el vuelo sólo sale los viernes, la fecha dada es un viernes). Esta operación se debe realizar en O(log N) promedio, donde N es la capacidad del vuelo.

- **void cancelacion(date fecha, int vuelo, string pasajero, estructura *E)**, en que dada una fecha (día/mes), el número de vuelo y el nombre de un pasajero, cancela una reservación hecha por el pasajero. Si el vuelo estaba lleno, el pasajero tenía lugar, y había gente en la lista de espera, agrega el primer pasajero en la lista de espera en la lista de pasajeros con lugar, y lo quita de la lista de espera. Se puede asumir que el vuelo existe en el día dado, y que el pasajero a dar de baja no está en la lista de espera para dicho viaje. Esta operación se debe realizar en O(log N) promedio, donde N es la capacidad del vuelo.

- **ListaVuelos salidas(int dia, Aeropuerto lugar, estructura E),** en que dado un día de la semana y un aeropuerto, devuelve la lista de vuelos de la compañía que salen en dicho día del aeropuerto. Esta lista debe estar ordenada por hora de salida. Esta operación se debe realizar en O(1) promedio.

- **ListaVuelos llegadas(int dia, Aeropuerto lugar, estructura E),** en que dado un día de la semana y un aeropuerto, devuelve la lista de vuelos de la compañía que llegan en dicho día al aeropuerto. Esta lista debe estar ordenada por hora de llegada. Esta operación se debe realizar en O(1) promedio.

- **void horarios(int vuelo, estructura E),** en que dado un número de vuelo imprime la lista de todos los vuelos semanales que tiene, ordenado por día de la semana, indicando hora y aeropuerto de salida, y hora y aeropuerto de llegada. Esta operación se debe realizar en O(1) peor caso.

- **boolean libre(date fecha, int vuelo, estructura E)**, en que dado una fecha y un número de vuelo, devuelve TRUE si el viaje de dicho vuelo a realizarse en la fecha dada tiene lugares libres y FALSE en caso contrario. Se puede asumir que el vuelo existe en el día dado. Esta operación se debe realizar en O(1) peor caso.

- **void pasajeros(date fecha, int vuelo, estructura E),** en que dado una fecha y un número de vuelo, imprime la lista de pasajeros con lugar reservado para el viaje de dicho vuelo a realizarse en la fecha dada. Esta lista debe estar ordenada alfabéticamente. Se puede asumir que el vuelo existe en el día dado. Esta operación se debe realizar en O(N) peor caso, donde N es la capacidad del vuelo.

- **void queViajes(string pasajero, estructura *E)**, en que dado el nombre de un pasajero, imprime la lista de viajes que tiene reservado. Los viajes deben estar ordenados por fecha de viaje. Para cada viaje, dicho listado debe incluir la siguiente información: número de vuelo, si tiene lugar o está en lista de espera, fecha de salida, aeropuerto de salida, fecha de llegada y aeropuerto de llegada. Un pasajero no tiene más de 10 viajes reservados, y se asume que los datos son consistentes (por ejemplo, que no tenga dos reservas en la misma hora en distintos

vuelos, o que esté dos veces en el mismo vuelo, etc.). Esta operación se debe realizar en O(1) promedio.

Se pide:
a. Diseñar estructuras de datos apropiadas para implementar eficientemente dichas operaciones, describiendo cómo se obtienen las cotas pedidas, y justificando sus decisiones.
b. Especificar la declaración de tipos en C.
c. Implementar la operación queViajes.

EJERCICIO 6 (R)

Una agencia de viajes maneja información acerca de países, sus ciudades (que constituyen destinos turísticos de interés), así como también de los paquetes turísticos que ofrece.

De un país se conoce su nombre, ubicación e idioma oficial. Un país se identifica por su nombre. De una ciudad se conoce su nombre y una descripción de los lugares que se pueden visitar en ella. Una ciudad se identifica por su nombre y por el nombre del país al que pertenece. Un país tiene como máximo M ciudades, y la agencia tiene como destino N países.

La agencia de viajes ofrece a sus clientes paquetes turísticos. Cada paquete se compone de un conjunto de estadías en ciudades (distintas) y se identifica por un número de dos cifras. Cada estadía es una terna (nombrePais, nombreCiudad, cantDias). Observe que una misma ciudad de un país, puede tener distinta cantidad de días de estadía para paquetes distintos.

Llamemos E a la estructura de datos utilizada para representar la realidad anterior. Se desea realizar las siguientes operaciones:

- **void infoCiudad(String nombrePais, String nombreCiudad, Estructura &E)**
  Dada la estructura E, el nombre de una ciudad nombreCiudad y el nombre del país nombrePais al que pertenece la ciudad, imprimir la información correspondiente a dicha ciudad. Esta operación debe realizarse en O(1) promedio.

- **void ciudadesPais(String nombrePais, Estructura &E)**
  Dada la estructura E y el nombre de un país nombrePais, imprimir las ciudades que pertenecen al mismo, ordenadas por su nombre alfabéticamente. Esta operación debe realizarse en O(M) promedio.
  Pre-condición: el país de nombre nombrePais pertenece a la estructura E.

- **void estadiasPaquete (int paquete, Estructura &E)**
  Dada la estructura E y un identificador de paquete, imprimir las ternas (nombrePais, nombreCiudad, cantDias) que están incluidas en el paquete. Esta operación debe realizarse en O(K) peor caso, siendo K la cantidad de ciudades incluidas en el paquete.

- **void altaCiudad(String nombrePais, String nombreCiudad, String dscLugares, Estructura &E)**
  Dada la estructura E, dar de alta la ciudad con nombre nombreCiudad perteneciente al país nombrePais y con la descripción de lugares dscLugares. Esta operación debe realizarse en O(logM) promedio.
  Pre-condición: el país de nombre nombrePais pertenece a la estructura E.

- **void agregarEstadia(int paq, String nombrePais, String nombreCiudad, int cantDias, Estructura &E)**
  Dada la estructura E, agrega al paquete con identificador paq una estadía de cantDias días en la ciudad nombreCiudad perteneciente al país nombrePais. Esta operación debe realizarse en O(1) promedio.
  Pre-condiciones: el paquete con identificador paq, el país de nombre nombrePais y su ciudad con nombre nombreCiudad pertenecen a la estructura E. A su vez, no existe una estadía en el paquete con identificador paq para la ciudad nombreCiudad del país de nombre nombrePais.

Se pide:

a. Diseñar una estructura (haga un dibujo con los comentarios pertinentes) que permita soportar las operaciones mencionadas con los órdenes de ejecución indicados. Explique detalladamente su estructura y justifique los órdenes de ejecución para cada operación.