PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMIA
Lista de Ejercicios: Listas, Pilas y Colas
(2013-2)

Horario 0582: prof. Fernando Alva

---

1. Sean `L` y `P` dos listas enlazadas que contienen números enteros ordenados ascendentemente. La función `ImprimePosiciones(L,P)` imprimirá los elementos en `L` que están en las posiciones especificadas por `P`. Por ejemplo, si `P` = 1,3,4,6, el primero, tercero, cuarto y sexto elemento en L deben ser impresos. Implemente la función indicada usando solo las operaciones básicas de listas (Extraído de [Wei95] - Traducción Libre).

2. (UVa 673 - Parentheses Balance) Una cadena con solo paréntesis ( ) y [ ] es correcta según las siguientes condiciones:

   - Si es una cadena vacía,
   - Si A y B son correctas, AB es correcta,
   - Si A es correcta, (A) y [A] son correctas.

   Escriba un programa que tome una secuencia de cadenas de caracteres del tipo indicado y verifique si son correctas o no. Por ejemplo:

   ```
   ([]) --> ES CORRECTA
   (([()]))) --> NO ES CORRECTA
   ([()[]()])() --> ES CORRECTA
   ```

3. Mientras que una *Pila* permite insertar y eliminar elementos en un solo extremo, y una *Cola* permite insertar en un extremo y eliminar en el otro, una ***deque*** (cola doblemente terminada) permite insertar y eliminar elementos en ambos extremos. Implemente 4 procedimientos $O(1)$ para insertar y eliminar elementos de ambos extremos de una *deque* utilizando listas enlazadas (Modificado de [CLRS09] - Traducción Libre).

4. Implemente el TAD *Conjunto* utilizando listas enlazadas. Considere las siguientes operaciones:

   - `Union(A,B,C)`: dados los conjuntos A y B, retorna el conjunto $C = A \cup B$
   - `Interseccion(A,B,C)`: dados los conjuntos A y B, retorna el conjunto $C = A \cap B$
   - `Diferencia(A,B,C)`: dados los conjuntos A y B, retorna el conjunto $C = A - B$
   - `Miembro(A,x)`: dados el objeto x (cuyo tipo es el tipo de elemento de A) y el conjunto A, retorna un valor booleano – True si $x \in A$ y False en caso contrario.
   - `CrearConjuntoVacio(A)`: convierte el conjunto A en un conjunto vacío.
   - `Insertar(A,x)`: inserta el objeto x (cuyo tipo es el tipo de elemento de A) en el conjunto A. El nuevo valor de $A = A \cup \{x\}$. Si x ya es un miembro de A, entonces la operación no altera el conjunto.
   - `Remover(A,x)`: remueve el objeto x (cuyo tipo es el tipo de elemento de A) del conjunto A. El nuevo valor de $A = A - \{x\}$. Si x no pertenece al conjunto A, entonces la operación no altera el conjunto.
   - `Min(A)`: retorna el menor valor del conjunto A. Por ejemplo: `Min`$(\{2,3,1\}) = 1$.
   - `Max(A)`: Similar a `Min(A)` solo que retorna el mayor elemento del conjunto.
   - `Igual(A,B)`: retorna True si y solamente si los conjuntos A y B contienen los mismos elementos.

5. Se desea realizar operaciones matemáticas básicas (suma, resta y multiplicación) en números enteros tan grandes que no pueden ser representados por ningún tipo de dado en C. Por lo tanto, la idea es representar cada número como una lista enlazada simple, donde cada nodo contenga un dígito. Implemente un TAD para poder manipular este tipo de números y realizar las operaciones básicas indicadas.

6. Un estacionamiento posee solo una línea para una máximo de 10 autos. Además, solo existe una entrada/salida en un lado de la línea. Si un cliente llega para recoger un auto que no es el más próximo a la salida, todos los autos que bloquean su camino son movidos, el auto del cliente es retirado y los otros autos son devueltos en el mismo orden en que estaban originalmente. Escriba un programa que procese un grupo de líneas de entrada. Cada una de ellas contiene una `L` para llegada y una S para salida, y un número identificador de auto. Se asume que los autos van a llegar y salir en el orden especificado en la entrada. El programa deberá imprimir un mensaje cada vez que un auto llega o sale. Cuando un auto llega, el mensaje debe especificar si existe una vacante en el estacionamiento. Si no existe alguna, el auto no entra en el estacionamiento. Cuando un auto sale, el mensaje debe incluir el número de autos que fueron movidos para que este pueda salir del estacionamiento (Extraído de [LAT90] - Traducción Libre).

7. Implementar un sistema de inscripción de personas para un evento teniendo en consideración lo siguiente:

   - Existen cuatro funcionalidades: registro de personas nuevas, actualización de datos de una persona, eliminación de una persona registrada y listado de personas registradas.
   - Para realizar el **registro**, cada persona debe informar obligatoriamente los siguientes datos: nombre, apellido y fecha de nacimiento (día, mes y año). Después, el nuevo registro debe ser agregado a la lista de personas registradas. Se debe verificar que no existan registros repetidos.
   - Para realizar la **actualización** de datos, el usuario debe indicar, por lo menos, algún dato informado en el registro (nombre, apellido y/o fecha de nacimiento). El sistema debe buscar el registro, mostrar el contenido y solicitar al usuario los datos que desea actualizar. En caso el usuario haya indicado un dato (o conjunto de datos) que no permite identificar de manera única el registro que se desea actualizar (por ejemplo, solo indica Apellido = "Alva" y existen varias personas con ese apellido, pero con otros datos diferentes), el sistema debe solicitar más datos hasta conseguir identificar a la persona que desea actualizar sus datos.
   - Para realizar la **eliminación**, el proceso es similar al realizado para la actualización. Después de identificar el registro que se desea eliminar, el sistema debe pedir confirmación al usuario antes de proceder a la eliminación del registro.
   - La funcionalidad de **listado** debe mostrar en pantalla la información de todas las personas registradas (una columna para cada dato) en orden ascendente por nombre. En caso de empate, se debe usar el apellido para identificar cuál registro va primero, y en caso de nuevo empate debe utilizar la fecha de nacimiento. Recuerde que no existen registros repetidos; por lo tanto, siempre será posible encontrar un orden entre los registros usando los tres datos.

   Utilizar listas enlazadas para implementar el sistema. Todas las implementaciones no deben poseer una complejidad mayor que $O(n)$.

8. (Extraído de Lista de Ejercicios 2004-2) Se desea sumar polinomios de la forma:

   $P_1 = 3x^2y + 5xy^3 + y - 12x^3 + 4y^2x^3$
   $P_2 = 6x + 9x^2y - 2y^2 + 12x^3$
   $P_1 + P_2 = 12x^2y + 5xy^3 + y + 4y^2x^3 + 6x - 2y^2$

   Se le pide:

   - Leer el coeficiente, exponenteX, exponenteY de cada uno de los términos que conforman un polinomio y almacenarlo en una lista enlazada cuyos nodos contengan dicha información para cada término.

- Implementar la suma de dos polinomios y almacenar el resultado en otra lista enlazada con la misma estructura.
- Imprimir el resultado de la suma de polinomios.

9. El sistema de atención en una agencia de un banco utiliza dos categorías para determinar el orden en que los clientes que llegan deben ser atendidos. Toda persona que posee una cuenta en el banco tiene la categoría A, caso contrario tiene categoría B. Se consideran las siguientes reglas para determinar cuál cliente debe ser atendido:

   - Si existe alguna caja vacía, un cliente (sea A o B) debe ser atendido inmediatamente.
   - Si llega un cliente de la categoría A, debe ser atendido la próxima vez que una caja quede vacía. Si existen otros clientes A esperando, el nuevo cliente deberá esperar a que ellos sean atendidos primero.
   - Un cliente de la categoría B solo podrá ser atendido si una caja está disponible y si no existe algún cliente A esperando.

   Implemente un programa para simular esta situación, considerando lo siguiente:

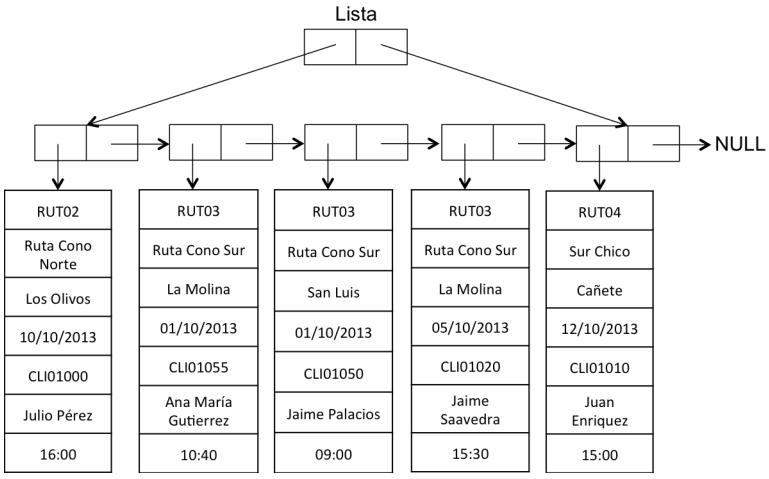
   - El programa deberá solicitar al usuario el número de clientes A, el número de clientes B y el número de cajas que serán considerados en la simulación.
   - El programa deberá escoger, aleatoriamente, el cliente que llega en un determinado momento al banco.
   - El programa deberá escribir en pantalla lo que está sucediendo en la simulación. Por ejemplo: *Llegó cliente A1*, *Atendiendo cliente B1 en Caja 2*, etc.

10. (Extraído de Lista de Ejercicios 2004-2) Se tiene un archivo binario "Rutas.dat", en donde se almacena la información sobre las rutas que sigue un vendedor durante sus visitas a los clientes con la finalidad de obtener una orden de pedido de ventas. Este archivo se encuentra ordenado ascendentemente por código de ruta y posee la siguiente estructura:

Nombre de Campo	**Tipo de Dato**	**Longitud**
Código de Ruta	String	5
Descripción de Ruta	String	50
Distrito	String	20
Fecha de Visita	String	10
Código Cliente	String	8
Nombre Cliente	String	50
Hora de Visita	String	5

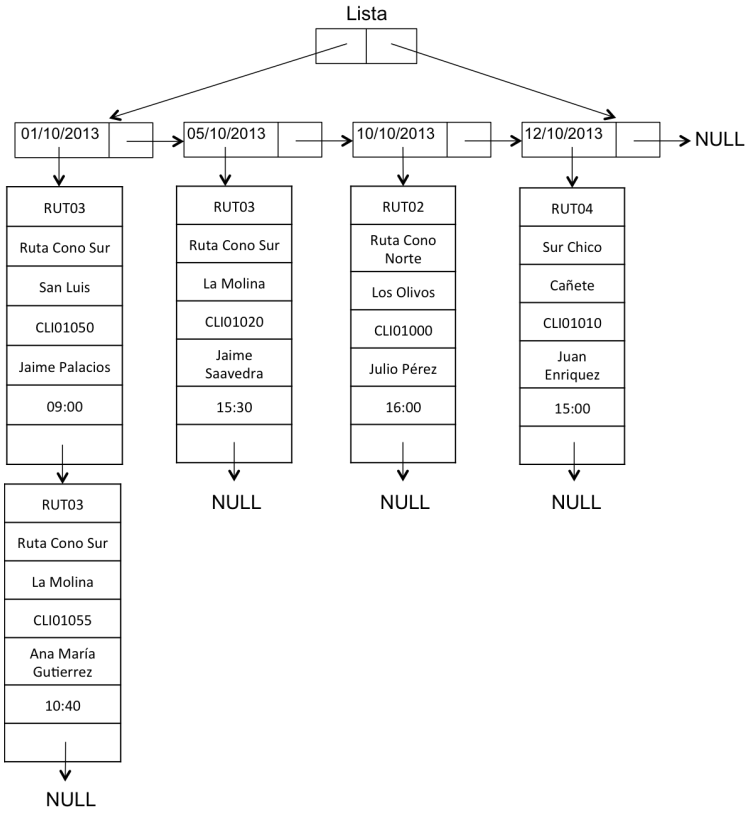
   Se le pide:

   - Leer el archivo binario "Rutas.dat" y generar una lista como la que se muestra a continuación:



El orden en el que se encuentran los elementos en la lista ordenada es el mismo orden que se sigue en el archivo binario.

- Usando la lista creada en el parte anterior, generar una lista de la forma siguiente:



Los nodos `Fecha de Visita` se encuentran ordenados ascendentemente. Además, dentro de cada nodo se encuentra ordenado por `Hora de Visita` ascendentemente.

- En base a la lista generada en la parte anterior, imprimir el resultado en un archivo de texto como el que se muestra a continuación:

```
Fecha: 01/10/2013
Relación de Visitas a Clientes
================================================================
09:00 A.M. - CLI01050 - Jaime Palacios
10:40 A.M. - CLI01055 - Ana María Gutierrez
================================================================

Fecha: 05/10/2013
Relación de Visitas a Clientes
================================================================
03:30 P.M. - CLI01020 - Jaime Saavedra
================================================================

Fecha: 10/10/2013
Relación de Visitas a Clientes
================================================================
04:00 P.M. - CLI01000 - Julio Pérez
================================================================

Fecha: 12/10/2013
Relación de Visitas a Clientes
================================================================
03:00 P.M. - CLI01010 - Juan Enriquez
================================================================
```

Referencias

[CLRS09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[LAT90] Yedidyah Langsam, Moshe J. Augenstein, and Aaron M. Tenenbaum. *Data Structures Using C and C++*. Prentice Hall, 1990.

[Wei95] Mark Allen Weiss. *Data Structures and Algorithm Analysis*. The Benjamim/Cummings Publishing Company, 1995.