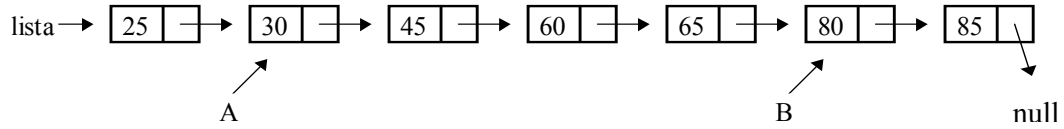
Trabajo Práctico N° 6: LISTAS

Dada la lista enlazada dinámica:



1. Dar los valores de las siguientes expresiones:

A.getInfo()	
B.getSig().getInfo()	
lista.getSig().getSig().getInfo()	

2. Verdadero o falso?

lista.getSig() == A	
A.getSig().getInfo() == 60	
B.getSig() == null	

3.Escribir la sentencia para cada una de las siguientes tareas:

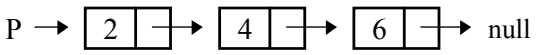
Hacer que lista apunte al nodo que contiene 45.	
Hacer que B apunte al último nodo de la lista.	
Hacer que lista apunte a una lista vacía.	

4. Utilizar las siguientes declaraciones para los ejercicios descriptos a continuación:

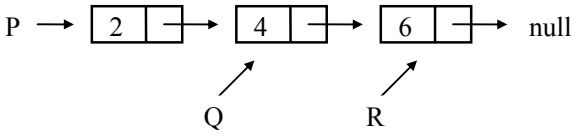
```
Nodo P, Q, R;
```

Mostrar en el diagrama lo que harían las sentencias:

a) P = P.getSig();
b) Q = P;
c) R = P.getSig();

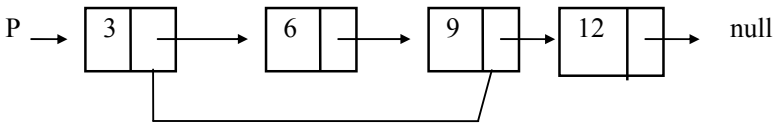


d) P.setInfo(Q.getInfo());
e) P.setInfo(Q.getSig().getInfo());
f) R.setSig(P);

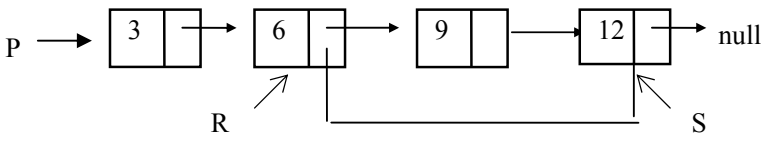


5. Escribir una sentencia para efectuar el cambio indicado en la línea de puntos

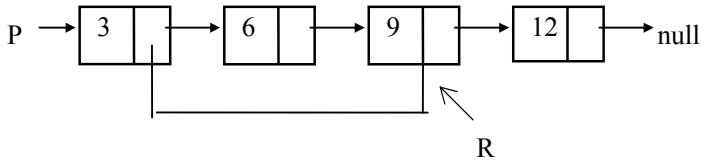
a)



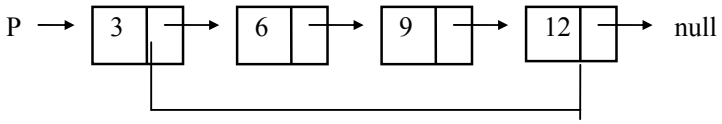
b)



c)



d)



Para todos los ejercicios definir los contratos que correspondan.

6. Implementar los siguientes métodos en la clase Lista y probar su correcto funcionamiento (los elementos son enteros y se insertan en forma ordenada)

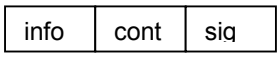
a) Escribir un método que busque un elemento en la lista y retorne verdadero o falso según corresponda.
b) Escribir un método que retorne la cantidad elementos de la lista.
c) Escribir un método que retorne la sumatoria de los elementos de la lista.
d) Escribir métodos que devuelvan el mayor y el menor valor, sin realizar comparaciones entre los valores.
e) Escribir un método que muestre la lista en forma inversa (usar una pila)
f) Escribir un método que muestre la lista en forma inversa (usar recursividad)
g) Escribir un método que invierta la lista.
h) Escribir un método que devuelva una copia de la lista.

7. Escribir la clase ListaDesordenada, con los métodos insertarPrimero (inserta siempre al principio) e insertarUltimo (inserta siempre al final) y los métodos restantes.(suprimir, mostrar, buscar, etc.)

8. Reescribir el Ej. 6 (b al h) pero con iteradores.

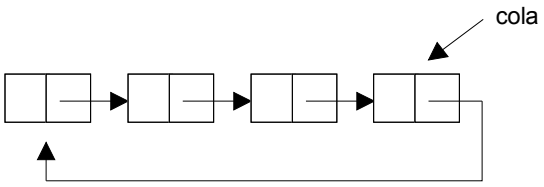
9. Escribir un programa que genere dos listas Pol1 y Pol2, las cuales almacenan polinomios. Luego generar un tercer lista SUMA, que contenga la suma de Pol1 y Pol2. Mostrar las tres listas.

10. Escribir un programa tipo menú, que permita generar una lista, insertar elementos, eliminar elementos y recorrer la lista. Podrán almacenarse elementos repetidos, pero para no desperdiciar espacio en memoria, se agrega un atributo al nodo:

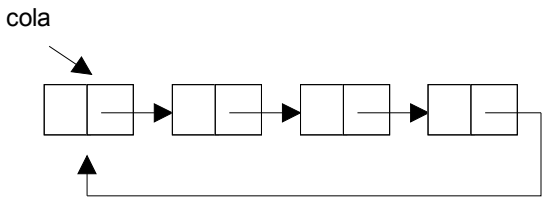


En el atributo info, se almacena el elemento. El atributo cont se incrementa en uno cada vez que se inserta una nueva ocurrencia de elemento y de la misma manera se decrementa cada vez que se realiza una eliminación del elemento. Cuando, cont queda en 0 se elimina el nodo de la lista.

11. Escribir para cada implementación graficada la clase Cola FIFO:
a. El puntero externo apunta al elemento final de la cola.



b. El puntero externo apunta al elemento frente de la cola.



12. Una lista enlazada (simple) circular, contiene elementos enteros, lista apunta al último nodo de la lista. Escribir un método que muestre los elementos positivos de la lista. Si no hay ninguno, mostrar el mensaje "NO HAY ELEMENTOS POSITIVOS".

13. El problema de José.
Consideremos un problema que puede ser resuelto en una forma directa mediante listas circulares. El problema se conoce con el nombre de José, y consiste en un grupo de soldados rodeados por una gran fuerza enemiga. No hay esperanza de victoria si no llegan refuerzos, y existe un solo caballo disponible para el escape. Los soldados se ponen de acuerdo en un pacto para determinar cuál de ellos debe escapar y solicita ayuda.
Forman un círculo y se escoge un número n al azar de un sombrero, igualmente se escoge el nombre de un soldado. Comenzando con el soldado cuyo nombre se ha seleccionado, comienzan a contar en la dirección del reloj al rededor del círculo. Cuando la cuenta alcanza n, este soldado es retirado del círculo y la cuenta empieza de nuevo, con el siguiente hombre. El proceso continúa de tal manera que cada vez que la cuenta alcanza n, se remueve un hombre del círculo. Un soldado que es removido del círculo por supuesto no se vuelve a contar. El último soldado que queda es el que debe tomar el caballo y escapar.
El problema es: dado un número n, el ordenamiento de los hombres en el círculo y el hombre a partir del que se comienza a contar, determinar el orden en el cual los hombres son eliminados del círculo y cual debe escapar. La entrada al programa es el número n y una lista de nombres que es el ordenamiento en el sentido de las manecillas del reloj en el círculo, comenzando con el hombre a partir del cual se debe comenzar a contar. La última línea de entrada contiene "FIN", indicando el final de la entrada. El programa debe mostrar los nombres de los soldados en el orden que han sido eliminados y el nombre de la persona que escapa.

Por ejemplo, supongamos que n es 3 y que hay 5 hombres denominados A, B, C, D y E. Contamos tres hombres partiendo de A, de tal manera que C es eliminado primero. Luego empezamos en D y contamos D, E y regresamos a A, de tal manera que A es eliminado. Después contamos B,D y E (C ya ha sido eliminado), se elimina E. Finalmente contamos B,D y B, se elimina B y D es el hombre que escapa.

14. Escribir la clase ListaDoble ordenada con los métodos básicos para su manipulación (constructor, insertar, eliminar, buscar, mostrar). Probar el funcionamiento.

15. Dadas las listas L1, L2 y L3 de enteros, donde:
- L1 es una lista enlazada simple
- L2 es una lista doblemente enlazada
- L3 es una lista enlazada circular

Escriba un programa que permita generar las tres listas (en forma ordenada). L2 y L3 se actualizan de acuerdo a L1.
a) Los valores de L1 que aparezcan en L2, serán eliminados de esta última.
b) Los valores de L1 que no estén en L3 se agregan a esta última y se eliminan a la vez de L1.
Nota: Usar una clase iteradora.

16. La clase Cola de Prioridad
El término *cola* sugiere la forma en que esperan ciertas personas u objetos la utilización de un determinado servicio. Por otro lado, el término *prioridad* sugiere que el servicio no se proporciona únicamente aplicando el concepto de cola FIFO, sino que cada persona u objeto tiene asociado una prioridad basada en un criterio objetivo.

Un ejemplo típico de organización formando colas de prioridades, es el sistema de tiempo compartido necesario para mantener un conjunto de procesos que esperan para trabajar. Los diseñadores de estos sistemas asignan ciertas prioridades a cada proceso.

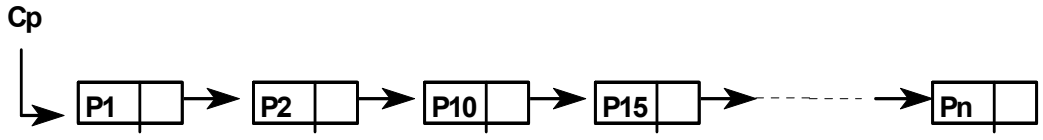
El orden en que los elementos son procesados y por tanto eliminados sigue estas reglas:

1. *Se elige la lista de elementos que tienen la mayor prioridad*
2. *En la lista de mayor prioridad, los elementos se procesan según el orden de llegada; en definitiva, según la organización de una cola: primero en llegar, primero en ser procesado.*

Las colas de prioridades pueden implementarse de dos formas: mediante una única lista o bien mediante una lista de colas.

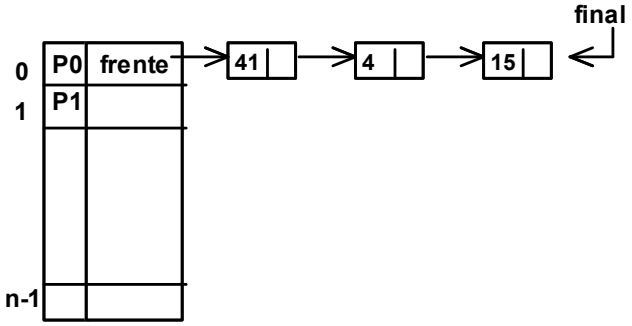
Implementación mediante una única lista
Cada proceso forma un nodo de la lista enlazada. La lista se mantiene ordenada por el atributo prioridad. La operación de añadir un nuevo nodo hay que hacerla siguiendo este criterio: *La posición de inserción es tal que la nueva lista ha de permanecer ordenada. A igualdad de prioridad se añade como último en el grupo de nodos de igual prioridad.* De esta manera la lista queda organizada de tal manera que un nodo X precede a uno Y si:

1. *Prioridad(X) > Prioridad(Y)*
2. *Ambos tienen la misma prioridad, pero X se añadió antes que Y.*



Los números de prioridad tienen el significado habitual: *a menor número mayor prioridad*.
Esta realización representa como principal ventaja que es inmediato determinar el siguiente nodo a procesar: siempre será el primero de la lista. Sin embargo, añadir un nuevo elemento supone encontrar la posición de inserción dentro de la lista, según el criterio expuesto anteriormente.

Implementación mediante una lista de n colas
Se utiliza una cola separada para cada nivel de prioridad. Cada cola puede representarse mediante una lista enlazada, con su *Frente y Final*. Para agrupar todas las colas, se utiliza un array de registros. Cada registro representa un nivel de prioridad, y tiene el frente y el final de la cola correspondiente. La razón para utilizar un array es que los niveles de prioridad son establecidos de antemano.



Las acciones más importantes al manejar una cola de prioridad son las de *añadir* un nuevo elemento, con una determinada prioridad al sistema, y la acción de *retirar* un elemento para su procesamiento. Estas acciones se expresan en forma algorítmica en la realización de *n* colas, como se muestra a continuación:

- Algoritmo para añadir nuevo elemento

Añade un elemento P que tiene un número de prioridad m.

1. *Buscar el índice en el array correspondiente a la prioridad m.*
2. *Si existe y es K, poner el elemento P como final de la cola de índice K.*
3. *Si no existe, crear una nueva cola y poner en ella el elemento P.*
4. *Salir.*

- Algoritmo para retirar un elemento

Retira el elemento frente de la cola que tienen máxima prioridad.

1. *Buscar el índice de la cola de mayor prioridad no vacía. Cola k.*

2. *Retirar y procesar el elemento frente de la cola k.*
3. *Salir.*

a) Codificar la clase Cola de Prioridad para la implementación mediante lista única.
b) Codificar la clase Cola de Prioridad para la implementación una lista de n colas.
c) Probar las implementaciones con el siguiente problema: Generar un programa tipo menú, que simule el funcionamiento de un sistema operativo, el cual debe ejecutar procesos de diferentes prioridades. Se debe poder generar y finalizar procesos. Los niveles de prioridad van de 0 a 9 y son dados en forma random, a cada proceso le corresponde un idproceso. El idproceso corresponde a un número interno, que se da correlativamente. (puede utilizar un contador). Cada vez que finaliza un proceso debe informarse la prioridad e idprocesos del mismo. Cuando se da de baja el sistema operativo, todos los procesos son retirados y se informa el total de procesos creados y total de procesos ejecutados.