PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

Algoritmia
Laboratorio 3
(Semestre 2014-1)

PARTE 1

1.- (8 puntos) Se pide implementar las siguientes funciones y/o procedimientos:

a) (3 puntos) Una función que recibe un puntero a una lista enlazada y devuelve el valor que más se repite en dicha lista.
(prototipo de la función *int moda(lista *l)*)

b) (3 puntos) Un procedimiento que recibe un puntero a una lista enlazada y debe eliminar los elementos repetidos dejándolos una sola vez en la lista.
(prototipo de la función *void eliminaRepetidos(lista *l)*)

c) (2 puntos) Un procedimiento que recibe un puntero a una lista enlazada de números enteros positivos y negativos (se excluye el 0) y debe modificar la lista de forma que los negativos se encuentren en el lado izquierdo y los positivos en el lado derecho. El programa debe tener estrictamente un tiempo O(*n*).
(prototipo de la función *void separaNP(lista *l)*)

Usted debe escribir sus programas principales de forma que validen los resultados esperados.

PARTE 2

2.- (7 puntos) En la especificación de listas dejada como tarea, el tipo de listas solo admite listas de elementos sobre un mismo tipo, como por ejemplo [1, -2, 3] para el caso de lista de enteros o [[1], [-2], [3]] en el caso de lista de listas de enteros, pero no es posible generar listas de la forma [-7, [8], 9] cuyas componentes son o bien elementos básicos o bien listas del mismo estilo. Implemente una librería estática para crear listas generales donde una lista general puede ser una lista vacía, o una lista formada por elementos de tipo entero o por listas generales. A continuación se especifican las operaciones.

- *lgvacia*, **(0,5 puntos)** crear una lista general vacía
- *lginserta*, **(0,5 puntos)** añade un elemento a la lista
- *lginsertalg*, **(1 punto)** añade un lista general a la lista general
- *lgizquierda*, **(1 punto)** devuelve el componente más a la izquierda de la lista
- *lgeliminaiz*, **(1 punto)** elimina el componente más a la izquierda de la lista
- *lgconcat*, **(1 punto)** concatena dos listas generales
- *lginvertir*, **(2 puntos)** invertir una lista general, invirtiendo a su vez cada una de sus componentes, por ejemplo: si L=[1, [2,3,4],[6,7],8] entonces *lginvertir(L)* devuelve [8,[7,6], [4,3,2], 1]

Debe escribir cada operación en un archivo por separado para luego empaquetarlas en una librería estática con nombre *liblistasg.a*

Corre por su cuenta elaborar un programa que pruebe su librería.

3.- (5 puntos) 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.

Cada nodo de la cola contiene: nombre del proceso (Proceso1,Proceso2, etc), prioridad (p1, p2, p3,..) y tiempo de llegada (0, 10, 23, ...).

Esta cola de prioridad cuenta con las siguientes operaciones:

- **(1 punto)** Crear una cola de prioridad vacía
- **(1 punto)** Añade un elemento a la cola de prioridad
- **(1 punto)** Retira el elemento de mayor prioridad o el más antiguo si tiene la misma prioridad
- **(1 punto)** Consulta el elemento que será servido
- **(1 punto)** Determinar si una cola de prioridad es vacía.

Debe escribir cada operación en un archivo por separado para luego empaquetarlas en una librería dinámica con nombre *libcolasp.so*

Queda a su criterio elegir emplear un sola lista o una lista de colas.

Corre por su cuenta elaborar un programa que pruebe su librería.

La estructura del directorio comprimido (*zip*) debe ser el siguiente:

```
código_de_alumno  --- |includes
                      |library  --| statics
                                  | dynamic
                      |sources
```

Pando, 27 de mayo de 2014.
Prof. Alejandro Bello