UNIVERSIDAD DE MÁLAGA

Programación de Sistemas y Concurrencia

Dpto. de Lenguajes y Ciencias de la Computación

Examen Final 20 de Junio 2017

APELLIDOS_____NOMBRE_____

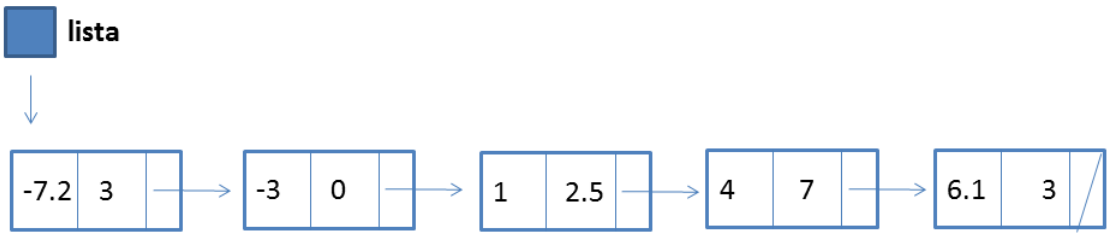
DNI _____ ORDENADOR _____GRUPO /TITULACIÓN_____

Bloque Programación C (3 puntos)

Una aplicación que dibuja funciones polinómicas utiliza una estructura basada en la siguiente información:

```
struct Punto {
  float x;
  float y;
}
```

Para la gestión de los diferentes puntos que se dibujarán en pantalla, se hace uso de una lista enlazada dinámica que permite optimizar el rendimiento de la aplicación. Es importante destacar que la información de los puntos debe estar ordenada en función de los valores de la coordenada x. En la siguiente figura puede observarse la forma en la que se almacena la información en memoria dinámica.



Implementa los siguientes procedimientos para la gestión de listas de puntos.

//Inicializa la lista de puntos creando una lista vacía
void crearLista(TLista *lista);

//Inserta el punto de forma ordenada (por el valor de la abscisa x) en la lista siempre que no esté repetida la abscisa. En ok, se devolverá un 1 si se ha podido insertar, y un 0 si no se pudo insertar. Nota: utiliza una función auxiliar para saber si ya hay un punto en la lista con la misma abscisa punto.x
void insertarPunto(TLista *lista, struct Punto punto, int *ok);

//Elimina de la lista el punto con abscisa x de la lista. En ok devolverá un 1 si se ha podido eliminar, y un 0 si no se pudo eliminar porque no existiera ese punto
void eliminarPunto(TLista *lista, float x, int *ok);

//Muestra en pantalla el listado de puntos
void mostrar_lista(TLista lista);

//Destruye la lista de puntos, liberando todos los nodos de la misma de memoria.
void destruir(TLista *lista);

//Lee el contenido del archivo binario de nombre nFichero, que contiene una secuencia de puntos de una función polinómica, y lo inserta en la lista.
void leePuntos(TLista *lista,char * nFichero);

En el campus virtual se encuentra el fichero Lista.h con la cabecera de los métodos que debes implementar, y un fichero binario Puntos.bin con los 6 puntos siguientes:

(35.25, 617.87)(36.94, 739.30)(24.77, -11.32)(73.46, 4770.97)(92.05, 7848.69)(25.00,0.00)

en el formato <p1x><p1y><p2x><p2y>… donde <p1x> y <p1y> son la representación en binario de las coordenadas x e y del primer punto, y así sucesivamente.

También hay un fichero Principal.c con dos métodos main (que debes renombrar) y que puedes utilizar para probar tu implementación.

Bloque Concurrencia (6 puntos)

Supón que, por seguridad, una guardería obliga a que siempre haya al menos un adulto por cada 3 bebés, es decir, que si nBebe y nAdulto son el número de bebés y adultos en la guardería, respectivamente, siempre debe cumplirse que **nBebe <= 3*nAdulto**. Por lo tanto, para implementar este sistema tenemos dos condiciones de sincronización

> **CS1**- Un bebé que quiere entrar en la guardería no puede hacerlo hasta que la condición nBebe <= 3*nAdulto sea cierta con él dentro.
>
> **CS2**- Un adulto que quiere salir de la guardería no puede hacerlo hasta que la condición nBebe <= 3*nAdulto sea cierta con él fuera de la guardería.

El esqueleto para resolver el ejercicio se encuentra en el campus virtual. Hay una clase Adulto, y un clase Bebe, que son las hebras que de manera ininterrumpida quieren entrar y salir de la guardería. La guardería está representada por un objeto de la clase Guardería que proporciona cuatro métodos:

public void entraBebe(**int** id) **throws** InterruptedException

public void saleBebe(**int** id) **throws** InterruptedException

public void entraAdulto(**int** id) **throws** InterruptedException

public void saleAdulto(**int** id) **throws** InterruptedException

Semáforos (3 puntos): Implementa la sincronización de los métodos de la clase Guardería utilizando **semáforos binarios**.

Monitores (3 puntos): Implementa la sincronización de los métodos de la clase Guardería utilizando **métodos sincronizados o locks**.