Programación II. Examen final. Mayo 2016. Bloque 1

Apellidos: _____Nombre:_____

Ejercicio 1. (2,5 puntos) Se desea diseñar e implementar una aplicación para utilizar un GPS. Imagina un modelo muy simplificado, donde SOLO se van a considerar las siguientes características:

- Una localización se representa mediante 2 coordenadas: latitud y longitud (números enteros).
- Un trayecto se representa mediante una localización origen y una localización destino.
- En el dispositivo GPS se pueden almacenar un máximo de 100 trayectos recientes.
- La aplicación principal del GPS debe proporcionar la funcionalidad siguiente:
  - Inicializar el dispositivo GPS
  - Crear una localización, a partir de una latitud y una longitud.
  - Obtener la latitud de una localización dada.
  - Obtener la longitud de una localización dada.
  - Asignar una latitud a una localización dada.
  - Asignar una longitud a una localización dada.
  - Crear un trayecto, a partir de su origen y su destino.
  - Obtener el origen de un trayecto dado.
  - Obtener el destino de un trayecto dado.
  - Asignar origen a un trayecto dado.
  - Asignar destino a un trayecto dado.
  - Añadir un trayecto al GPS.
  - Eliminar un trayecto del GPS.
  - Obtener el nº de trayectos almacenados en el GPS.
  - Imprimir todos los trayectos almacenados en el GPS: imprimir el nº de trayectos almacenados y, para cada uno de ellos, su longitud y su latitud.

a) (0.2) ¿Cuántos y qué nombre darías a los Tipos Abstractos de Datos necesarios para representar lo descrito anteriormente? Los tipos status y boolean se encuentran ya definidos (no se consideran aquí)

Nº TADs: 3 Nombres TADs: GPS, Trayecto y Localización.

b) (0.5) Escribe el código en C de las Estructuras de Datos (EdD) necesarias para implementar los TADs que has respondido en la pregunta anterior, y crea los nuevos tipos de datos correspondientes.

```
typedef struct _Localizacion{
    int latitud;
    int longitud;
}Localizacion;

typedef struct _Trayecto{
    Localizacion *origen;
    Localizacion *destino;
} Trayecto;

typedef struct _GPS{
Trayecto *trayectos[100];
    int num_tray;
} GPS;
```

c) (0.8) Escribe el prototipo de la primitivas:

- creaLocaliz: crea una localización a partir de dos parámetros que expresan su latitud y longitud.

```
Localizacion * creaLocaliz (int latitud, int longitud);
```

- creaTrayecto: crea un trayecto a partir de dos localizaciones, la de origen y la de destino.

```
Trayecto *creaTrayecto (Localizacion *origen, Localizacion *destino);
```

- numTrayectosDeGPS: dado un GPS, obtiene/devuelve el nº de trayectos almacenados en el mismo.

```
int numTrayectosDeGPS (GPS *gps);
```

- imprimeTrayectosDeGPS: dado un fichero y un GPS, imprime en el fichero los trayectos almacenados en el GPS.

```
void imprimeTrayectosDeGPS (GPS *gps, FILE *f);
```

a) (0,5) Dada la implementación en C de los TADs mediante las estructuras que has propuesto en el ejercicio anterior, proporciona el código C (con control de errores) de la siguiente función:

```
/*============================================================================
Nombre: anyadeTrayectoAlGPS
Descripción: Añade un trayecto a la lista de trayectos almacenada en el GPS.
Recibe:  origen: localización origen que tendrá el trayecto a añadir al GPS
         destino: localización destino que tendrá el trayecto a añadir al GPS
         gps: GPS al que se añadirá el nuevo trayecto.
Devuelve: OK si añade correctamente un nuevo trayecto, ERROR si no.
Modifica: Si devuelve OK, en GPS se ha añadido el trayecto. Si devuelve ERROR, nada.
============================================================================*/
```

¿En qué fichero debe implementarse este código? gps.c

```
Status anyadeTrayectoAlGPS (Localizacion *origen, Localizacion *destino, GPS *gps){
  if (!origen || !destino || !gps) {
    return ERROR;
  }
  if (gps->num_tray >= 100) return ERROR;
  Trayecto *trayecto = NULL;
  if ((trayecto = creaTrayecto(origen, destino)) != NULL) {
    gps->trayectos[gps->num_tray] = trayecto;
    gps->num_tray++;
    return OK;
  }
  return ERROR;
}
```

0,5

b) (0,5) Implementa en C (sin control errores) una función main que cree un GPS y un trayecto con origen en la localización (20, 20) y destino en la localización (30,30), almacene el trayecto en el GPS e imprima la información almacenada en el GPS.

```
int main (){
    GPS *g;
    FILE *f;
    Localizacion *o, *d;
    o = localizacion_creaLocaliz(20,20);
    d = localizacion_creaLocaliz(30,30);
    g = GPS_crearGPS();
    GPS_anyadeTrayectoAlGPS(g, o, d);
    GPS_imprimeTrayectosDeGPS(g, stdout);
    GPS_eliminarTrayectosDeGPS(g);
    Localizacion_eliminarLocalizacion(o);
    Localizacion_eliminarLocalizacion(d);
    return 0;
}
```