UNIVERSIDAD DE MÁLAGA

Departamento de Lenguajes y Ciencias de la Computación

LABORATORIO DE PROGRAMACIÓN

Examen Ordinario de Septiembre

1ºGestión Curso 2010/2011

23/09/2011 a las 9:30

APELLIDOS_____ NOMBRE_____

DNI _____ ORDENADOR _____ GRUPO (A/B)_____

El Instituto Astrofísico de Canarias nos pide que desarrollemos una aplicación para gestionar las peticiones de observación que se producen en un día en el Gran Telescopio de Canarias (GTC). De cada petición de observación debemos almacenar el nombre del investigador (una cadena de caracteres), la hora de inicio y de fin de la observción (un valor de tipo `THora`) y el instrumento de observación utilizado (un enumerado), que puede ser uno de estos cinco: `osiris`, `canaricam`, `circe`, `emir` y `frida`. Estas peticiones se representarán con valores de tipo `TPeticion` y serán almacenados en una lista (tipo `TLista`) ordenadas por hora de inicio.Se pide realizar una aplicación que presente las siguientes opciones de menú:

```
Nombre: (Apellidos, Nombre)          Curso: 1º
Especialidad: Gestión                Grupo: A/B
Puesto: número de ordenador          Fecha: 23/09/2011

OBSERVACIONES EN EL GTC
=======================
A. Insertar Petición.
B. Listar Peticiones.
C. Cargar Peticiones desde Fichero.
D. Salvar Peticiones a Fichero.
E. Buscar Petición.
X. Salir del Programa

Introduzca Opción:
```

A. **`Insertar Petición.`**Se pedirá desde teclado el nombre del investigador, la hora de inicio de la observación, la hora de fin de la observación y el instrumento utilizadoy se insertará en la estructura de forma ordenada. Si ya existiera unapetición de observación que ocupara parte del tiempo de observación solicitado no se insertaría la petición y se informaría del error. Si la hora de finalización es anterior o igual a la de inicio, se le pedirá al usuario que introduzca la franja horaria de nuevo.

`Ejemplo:`

```
Investigador: Antonio López
Hora de inicio: 9:30
Hora de fin:10:30
Instrumento: circe
Operación completada satisfactoriamente.
```

B. **`Listar Peticiones.`** Se mostrará por pantalla un listado de todas las peticionesordenadas por hora de inicio.

`Ejemplo:`

```
Listado de Peticiones
=====================
Investigador: Antonio López
Hora de inicio: 9:30
Hora de fin: 10:30
Instrumento: circe

Investigador: Stephen Hawking
Hora de inicio: 10:30
Hora de fin: 14:30
Instrumento: osiris

Investigador: Roger Penrose
```

```
Hora de inicio: 16:00
Hora de fin: 17:15
Instrumento: canaricam
```

C. **Cargar Peticiones desde Fichero.** Se pedirá el nombre de un fichero de texto y se insertará en la estructura toda la información existente con el formato adjunto (sin eliminar de las estructura las peticionas que ya tuviera almacenadas). Se deberá informar de cualquier error relativo al manejo de ficheros o de memoria. En caso de que alguna petición no se pueda incluir por existir otra en la estructura que se solape en franja horaria se ignorará el error.Formato del fichero:

```
<INVESTIGADOR>@<HORA_INICIO>:<MINUTOS_INICIO>@<HORA_FIN>:<MINUTOS_FIN>@<INSTRUMENTO><ENTER>
<INVESTIGADOR>@<HORA_INICIO>:<MINUTOS_INICIO>@<HORA_FIN>:<MINUTOS_FIN>@<INSTRUMENTO><ENTER>

   . . . tantas líneas como peticiones
<INVESTIGADOR>@<HORA_INICIO>:<MINUTOS_INICIO>@<HORA_FIN>:<MINUTOS_FIN>@<INSTRUMENTO><ENTER>

Ejemplo: peticiones.txt
   Antonio López@9:30@10:30@circe
   Stephen Hawking@10:30@14:30@osiris
   Roger Penrose@16:00@17:15@canaricam
```

D. **SalvarPeticionesa Fichero.** Se pedirá el nombre de un fichero de texto y se salvará en dicho fichero la información contenida en la estructura de datos siguiendo el formato indicado en el apartado anterior.

E. **BuscarPetición.** Se pedirá una hora por teclado y se deberá mostrar la petición almacenada en la estructura de datos que se está realizando en la hora indicada. Si no hay ninguna petición planificada para esa hora se indicará con un mensaje. Si la hora se corresponde con el fin de una petición y el inicio de otra, se mostrará la petición que comienza.

Ejemplo:

```
Hora: 13:00

Investigador: Stephen Hawking
Hora de inicio: 10:30
Hora de fin: 14:30
Instrumento: osiris

Operación completada satisfactoriamente.
```

X. **Salir del Programa**. Se pedirá confirmación de salida.

Módulos a Implementar:(LA DEFINICIÓN DE LOS MÓDULOS NO SE PODRÁ MODIFICAR)

- `MError`: Manejo de errores.
  - Define el tipo `Terror`como un enumerado con los posibles errores generados.
  - Define los siguientes PROC/FUNC:
    - `MostrarError(error)`: Muestra el error por pantalla.
- `MHora`:
  - Define la constante`DP (':')`
  - Define el tipo`THora como un registro con dos campos (hora y minutos) de tipo natural` y el enumerado `TOrdenHora`
  - Define los siguientes PROC/FUNC:
    - `ComparaHora(hora1,hora2)`: Compara las horas que se le pasan por paráetro y devuelve un valor de tipo `TOrdenHora` con tres posibles valores, `menor`, `igual` y `mayor`, que indican si `hora1` es menor, mayor o igual que `hora2`, respectivamente.
    - `LeeHora(hora)`: Lee una hora desde teclado.
    - `EscribeHora(hora)`: Escribe una hora por pantalla.
    - `LeeHoraFichero(fich,hora)`: Lee una hora desde un fichero de texto cuyo manejador se le pasa como parámetro.
    - `EscribeHoraFichero(fich,hora)`: Escribe una hora en un fichero de texto cuyo manejador se le pasa como parámetro.
- `MPeticion`:
  - Define el tipo`TPeticion`y`TInstrumento`y las constantes `ARROBA ('@')` y `ENTER ('\n')`
  - Define los siguientes PROC/FUNC:
    - `Instrumento_a_Cadena(c)`: Devuelve la cadena que representa a un instrumento.
    - `Cadena_a_Instrumento(s,i,ok)`: Convierte, si es posible, una cadena en el instrumento al que representa.
    - `MostrarPeticion(p)`: Muestra por pantalla una petición.
    - `LeePeticion(p)`: Lee una petición desde el teclado.

- `EscribePeticionFichero(fich,p)`: Escribe la petición en el fichero de texto cuyo manejador se le pasa como parámetro.
- `LeePeticionFichero(fich,p)`: Lee la petición desde el fichero de texto cuyo manejador se le pasa como parámetro.
- `Investigador(p)` **:** Devuelve el nombre del investigador que realizó la petición.
- `HoraInicio(p)` **:** Devuelve la hora de inicio de la petición.
- `HoraFin(p)` **:** Devuelve la hora de fin de la petición.
- `Instrumento(p)` **:** Devuelve el instrumento necesario para la observación.

❖ `MLista`:

- Define los tipos`TLista` y `TNodo`
- Define los siguientes PROC/FUNC:
  - `CrearLista()`: Crea una Lista Vacía.
  - `ListaVacia(l)`: Comprueba si una lista está vacía.
  - `ListaLlena(l)`: Comprueba si una lista está llena.
  - `InsertarLista(l,p,error)`: Inserta una petición en la lista.
  - `MostrarLista(l)`: Muestra una lista por pantalla.
  - `BuscarLista(l,h,p,error)` **:** Devuelve la petición de observación que se lleva a cabo en la hora indicada.
  - `EscribeListaFichero(l,nomFich,error)` **:** Escribe la lista de peticiones en un fichero de texto cuyo nombre se le pasa como parámetro.
  - `LeeListaFichero(l,nomFich,error)` **:** Lee las peticiones almacenadas en el fichero de texto cuyo nombre se le pasa como parámetro y las almacena en la lista.
  - `DestruirLista(l)`: Libera toda la memoria reservada por la lista.

NOTAS

1. El directorio de trabajo será `Mis Documentos\LPGSEP11`

2. OPCIONES A, B y C (5 puntos.)

3. OPCIONES D, Ey calidad de la implementación (opciones para subir nota) (5 puntos)

4. **El uso de detalles de implementación fuera del módulo de implementación** del mismo será **CAUSA de SUSPENSO**, es decir, en el programa principal o en el módulo MLista no se podrá hacer cosas tipo: `l=NULL, peticion.HoraInicio`, etc.

5. **Añadir procedimientos o funciones a la definición de un módulo** será **CAUSA de SUSPENSO**

6. Todo **PROGRAMA QUE NO COMPILE** o tenga **efectos laterales** se considerará **SUSPENSO**.

7. No se distinguirá entre mayúsculas y minúsculas.

8. Se recomienda y valora el tratamiento de errores y la buena descomposición del programa principal en procedimientos y funciones, así como el uso de procedimientos y funciones auxiliares dentro de la implementación de los módulos cuando éstas sean necesarias.

9. Recordad que todo ficherode cabecera debe llevar su pragma #ifndef .... #endif. Por ejemplo:

```
#ifndef _MError_h_
#define _MError_h_
      . . .
#endif
```