UNIVERSIDAD DE MÁLAGA

Departamento de Lenguajes y Ciencias de la Computación

LABORATORIO DE PROGRAMACIÓN

E.T.S.I. Informática- 1º GESTION. XX-SEPTIEMBRE-2008

APELLIDOS_____NOMBRE_____

DNI _____ ORDENADOR_____GRUPO (A/B/C)_____

RENFE nos ha encargado un programa para simular el recorrido de un tren de alta velocidad. Los trenes se modelarán utilizando listas enlazadas en la que cada nodo representa un vagón del tren.
Un vagón viene dado por una array de asientos (con MAX posiciones). Cada asiento tendrá un registro con un pasajero y un campo lógico que indique si el asiento está vacío o no.
Un pasajero se representa como un registro con los siguientes campos: nif, billete y kilos de equipaje. Cada billete contendrá el origen, el destino, el número de asiento asignado y el tipo (ventanilla o pasillo).

Fig. 1 Representación de un tren con sus vagones y sus listas de pasajeros.

El programa deberá presentar un menú como el siguiente. Antes de mostrarlo por primera vez se preguntará al usuario el número de vagones y se creará un tren de ese tamaño.

```
 Nombre: (Apellidos, Nombre)    Curso: 1º
 Especialidad: Gestión          Grupo: A/B/C
 Puesto: número de ordenador    Fecha: xx/09/2008

     RENFE
     =====
 A. Subir pasajero al tren.
 B. Mostrar pasajeros del tren.
 C. Salvar datos de pasajeros en un fichero de texto.
 D. Leer lista de pasajeros de un fichero de texto.
 E. Parada en estación: bajada de viajeros.
 F. Buscar pasajero en el tren por su DNI.
 G. Mostrar recorrido.
 X. Salir del Programa.

Introduzca Opción: _
```

DESCRIPCIÓN DE OPCIONES:

- **Mínimo Obligatorio para Aprobar para quien tenga superado el trabajo en clase** (3 puntos): correcta la definición de tipos, la modularización y funcionar **CORRECTAMENTE** las Opciones A, B, C, D y X del menú.
  IMPORTANTE: Para aprobar no es necesario completar todas las funciones de los módulos, tan sólo aquellas (resaltadas en negrita) que sean necesarias para completar las opciones del menú obligatorias.
- **Mínimo Adicional Obligatorio para Aprobar para quien NO tenga superado el trabajo en clase** (2 puntos): funcionar **CORRECTAMENTE** las Opciones E y F del menú.

A. Subir pasajero al tren.
Se leerá desde teclado toda la información de un pasajero que será insertado en el vagón correspondiente según el número de asiento de su billete. Para ello se ha de tener en cuenta que cada vagón tiene un número máximo de asientos y que la numeración empieza en 1 a partir del primer vagón. Por ejemplo si tenemos un tren de cuatro *vagones* de *20* plazas, los números de asiento irían *1 al 20, del 21 al 40, del 41 al 60 y del 61 al 80.* Si un pasajero tiene el asiento nº 52 en su billete iría sentado en el 3er vagón.
Si los kilos de equipaje son más de cien se debe indicar un error por exceso de equipaje y el pasajero no podrá subir al tren. Si ya existe un pasajero con el mismo DNI se actualizarán sus datos. Supondremos que los biletes han sido correctamente emitidos y que NO se dará el caso de que el asiento ya esté ocupado. Ejemplo:

```
DNI                : 123456
Billete
      Origen : Málaga
      Destino: Barcelona
      Asiento: 132
      Tipo    : ventanilla
Kilos de equipaje: 50
```

B. Mostrar pasajeros del tren. El usuario deberá mostrar por pantalla los vagones del tren y para cada vagón imprimirá la lista completa de pasajeros que viajan en él.

C. Salvar datos de pasajeros en un fichero de texto. Se almacenarán en un fichero de texto los datos de todos los pasajeros del tren. El fichero tendrá el siguiente formato:

```
DNI Origen#Destino#Asiento Tipo#Kilos
DNI Origen#Destino#Asiento Tipo#Kilos
...
```

La última línea acabará con un retorno de carro.

Ejemplo:

```
123456 Málaga#Barcelona#132 ventanilla#50
123457 Córdoba#Madrid#34 pasillo#10
```

D. Parada en estación: bajada de viajeros.
Se eliminarán del tren todos los pasajeros cuyo destino coincida con el nombre de una estación que se le pasa como parámetro.

E. Leer lista de pasajeros de un fichero de texto
Leerá desde un fichero de texto (con el formato descrito en la opción C) los datos de pasajeros que se insertará en un tren que se le pasa como parámetro.

F. Buscar pasajero en el tren por su DNI. Se devolverán todos los datos del pasajero cuyo DNI se pasa como parámetro.

G. Mostrar recorrido.
En un fichero de nombre *recorrido.txt* se almacenan los nombres de todas las estaciones por las que pasará un tren desde su salida hasta su llegada. Cada nombre de una estación irá en una línea del fichero. Por cada una de ellas hay un fichero de nombre *estación.txt* (con el mismo formato del

descrito en el apartado C) que contiene los datos de los pasajeros que se suben al tren en dicha estación.
Partiendo de estos ficheros el programa deberá mostrar por pantalla la evolución del tren a lo largo de su recorrido indicando en cada estación qué pasajeros bajan y cuáles suben del tren.
Ejemplo
Ficheros:

recorrido.txt	*Málaga.txt*	*Bobadilla.txt*	*Córdoba.txt*	*Madrid.txt*
Málaga *Bobadilla* *Córdoba* *Madrid*	11 Málaga#Bobadilla#13 pasillo#50 99 Málaga#Madrid#34 pasillo#10 12 Málaga#Córdoba#50 pasillo#23	1 Bobadilla#Córdoba#12 pasillo#20 9 Bobadilla#Madrid#1 pasillo#10	…	…

Salida del programa:

```
Málaga:
      Bajan: ---------------------
      Suben:
            DNI: 11  Asiento: 13
            DNI: 99  Asiento: 34
            DNI: 12  Asiento: 50
Bobadilla:
      Bajan:
            DNI: 11  Asiento: 132
      Suben:
            DNI: 1   Asiento: 12
            DNI: 9   Asiento: 1
Córdoba:
      Bajan:
            DNI: 12  Asiento: 50
            DNI: 1   Asiento: 12
      Suben:
            …
Madrid:
      Bajan:
            DNI: 11  Asiento: 13
            DNI: 9   Asiento: 1
            …
      Suben: --------------------
```

X. Salir del Programa. En esta opción se pedirá confirmación de salida. Si es afirmativa, se terminará el programa, liberando todos los recursos que hayan sido reservados y si es negativa, se volverá al menú principal.

Módulos a Implementar:

- **`Programa principal`** (`viaje.cpp`)
- `MCadena` (`MCadena.h, MCadena.cpp`): Módulo de manejo de cadenas.
  - Define las constantes necesarias y el tipo `TCadena` como un array de 80 caracteres.
  - Define el enumerado `TCompara` con los valores: `Menor, Igual, Mayor`.
  - Define las funciones `CopiaCadena`, `ComparaCadena y ConcatenaCadena`.
- `MPasajero` (`MPasajero.h, MPasajero.cpp`): Módulo de manejo de pasajeros
  - Define las **constantes necesarias**.
  - Define los tipos **`TNatural`** (entero sin signo), **`TTipoAsiento`** (enumerado), **`TBillete`** (registro origen, destino, asiento y tipo asiento), y **`TPasajero`** (registro con el dni, el billete y los kilos de equipaje) y **`TPasaje`** (array de pasajeros).
  - Define la función **`SUC_TipoAsiento`**: retorna el sucesor del `TTipoAsiento` que se le pasa.
  - Define la función **`TipoAsiento_a_Cadena`**: convierte un `TTipoAsiento` a cadena.
  - Define la función **`Cadena_a_ TipoAsiento`**: convierte una cadena a `TTipoAsiento`.
  - Define la Función **`LeerTipoAsiento`**: retorna un `TTipoAsiento` leído desde teclado.
  - Define la función **`LeerBillete`**: lee un registro `TBillete` desde teclado.
  - Define la función **`EscribirBillete`**: escribe por pantalla un registro `TBillete`.

- Define la función **LeerPasajero:** lee un registro `TPasajero` desde teclado.
- Define la función **EscribirPasajero**: escribe por pantalla un registro `TPasajero`.
- Define la función **InsertarPasajero**: almacena un registro `TPasajero` en una posición `i` de un array `TPasaje`.
- Define la función **DevolverPasajero**: devuelve un registro `TPasajero` almacenado una posición `i`.
- Define la función `LeerPasajeroFicheroTXT`: lee un registro `TPasajero` desde un fichero de texto cuyo manejador se le pasa como parámetro.
- Define la función **EscribirPasajeroFicheroTXT:** escribe un registro `TPasajero` en un fichero de texto cuyo manejador se le pasa como parámetro.
- Define la función `Origen` que devuelve el origen de un billete.
- Define la función `Destino` que devuelve el destino de un billete.
- Define la función **Asiento** que devuelve el asiento de un billete.

- `MVagon (MVagon.h, MVagon.cpp:` Módulo para el manejo del `TVagon`
  - Define las **constante necesarias** `y` los tipos **TErrorVagon** `y` **TVagon**.
  - Define la función **CrearVagon**: Crea un vagon e inicializa todos los aseintos como vacíos.
  - Define la función **VagonVacio**: Nos dice si un `vagón` está vacío, es decir si no hay pasajeros en ninguno de sus vagones.
  - Define la función **VagonLleno**: Nos dice si un `vagón` está lleno, es decir, si todos sus vagones están completos.
  - Define la función **OcuparAsiento**: Inserta un pasajero en el vagón que le corresponde y pone su asiento como ocupado.
  - Define la función **MostrarVagon**: Muestra todos los pasajeros del vagón.
  - Define la función `LiberarAsiento`: marca un asiento del vagón como desocupado.
  - Define la Función `EliminarPasajerosDestinoVagón`: Elimina (si existen) todos los pasajeros cuyo destino coincida con el nombre de la ciudad que se pasa como parámetro.
  - Define la Función `BuscarPasajeroVagón`: Retorna (si existe) un pasajero dado su nif.
- `MTren (MTren.h, MTren.cpp)`: Módulo de manejo del `TTren`.
  - Define las **constante necesarias** `y` los tipos **TErrorTren** `y` **TTren**.
  - Define la función **CrearTren**: Crea un tren con *n* vagones de MAX asientos cada uno.
  - Define la función **InsertarVagon** que inserta un vagón en una posición dada.
  - Define la función **TrenVacío**: Nos dice si un `tren` está vacío, es decir si no hay pasajeros en ninguno de sus vagones.
  - Define la función **TrenLleno**: Nos dice si un `tren` está lleno, es decir, si todos sus vagones están completos.
  - Define la función **SentarPasajero**: Inserta un pasajero en el vagón que le corresponde.
  - Define la función **MostrarTren**: Muestra todos los pasajeros del tren.
  - Define la función `EliminarPasajero`: Elimina (si existe) un pasajero del tren dado su número de asiento.
  - Define la Función `EliminarPasajerosDestino`: Elimina (si existen) todos los pasajeros cuyo destino coincida con el nombre de la ciudad que se pasa como parámetro.
  - Define la Función `BuscarPasajero`: Retorna (si existe) un pasajero dado su dni.
  - Define la Función **DestruirTren**: Destruye un tren liberando toda la memoria.
  - Define la Función `CargarTrenFichero`: Inserta todos los pasajeros que haya en un fichero cuyo nombre se pasa como parámetro y con el formato indicado anteriormente.
  - Define la Función **SalvarTrenFichero**: Almacena todos los pasajeros de un tren en un fichero cuyo nombre se pasa como parámetro en el formato indicado anteriormente.

NOTAS

1. Es obligatorio trabajar en el directorio **C:\LPSEP08**.
2. **El uso de detalles de implementación de la lista fuera del módulo de implementación** de la misma (`MTren.cpp`) será **CAUSA de SUSPENSO**, es decir, en el programa principal no se podrá hacer cosas tipo: `tren= NULL`, `ptr=ptr->sig`, etc.
3. **Añadir procedimientos o funciones a la definición de un módulo** será **CAUSA de SUSPENSO**

4. Todo **PROGRAMA QUE NO COMPILE** o tenga **efectos laterales** se considerará **SUSPENSO**.
5. No se distinguirá entre mayúsculas y minúsculas.
6. 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 estas sean necesarias.