Universidad de Granada
Departamento de Ciencias de la Computación e Inteligencia Artificial

Fundamentos de Programación (2019/2020)
1º GII / GII-M / GII-ADE
Conv. Extraordinaria. Examen de prácticas - 30 de Enero de 2020

DECSAI

Normas para la realización del examen: **Duración: 1.5 horas**

- **El profesor le indicará en clase cómo ha de entregar el fichero cpp que resuelve el problema planteado en este examen.**

◁ Ejercicio 1 ▷ Agenda [4 puntos]

El objetivo de este ejercicio es trabajar con una agenda de contactos. En clase se proporcionará un fichero `Agendas.cpp` sobre el que puede trabajar.

Suponga la siguiente clase `Contacto`:

Contacto
- string nombre - string apellidos - string email
+ Contacto() + string Nombre() + string Apellidos() + string Email() + void SetNombre(string nombre_contacto) + void SetApellidos(string apellidos_contacto) + void SetEmail(string email_contacto) + string ToString()

Table 1: Clase disponible y métodos que **NO hace falta implementar**.

Defina la clase `Agenda` como un conjunto de datos de tipo `Contacto`. La clase tendrá que proporcionar, al menos, métodos para recuperar un contacto dado un índice (desde 0), para añadir un contacto y para consultar el número de contactos almacenados en la agenda. Se pide construir métodos para realizar las siguientes tareas:

- Ordenar los datos de una agenda según el apellido -> `Método Ordena`
- Comprobar si dos agendas son iguales -> `Método EsIgual`
- Añadir todos los contactos de una agenda a la otra -> `Método AniadeAgenda`
- Fusionar de forma ordenada dos agendas: el resultado será una agenda ordenada por apellido -> `Método Fusiona`

  Obligatoriamente se pide que el método se implemente de la siguiente forma:

  1. Ordene cada agenda.
  2. Mezcle de forma ordenada los contactos de ambas agendas.

Construya un programa principal que lea los contactos de dos agendas. Para ello, utilice la clase `LectorAgendas` disponible en el fichero `Agendas.cpp`. El método `Lee()` lee los datos desde el periférico de entrada de datos por defecto y devuelve una agenda. Para su información, los datos de nombre, apellidos e email se introducen separados por un retorno de carro y el final de los datos de una agenda viene marcado por el carácter `'#'`. En el fichero `Agendas.cpp` puede encontrar el código de la clase `LectorAgendas` y un ejemplo con los contactos de dos agendas.

Después de leer las dos agendas, se pide lo siguiente:

- Construya una tercera agenda `fusion` fusionando de forma ordenada las dos agendas anteriores (método `Fusiona`)
- Construya una cuarta agenda `suma`, que contenga los datos de la primera añadiéndole los de la segunda a través del método `AniadeAgenda`. Ordene los contactos de `suma` ejecutando el método `Ordena` y compruebe que es igual a la agenda `fusion` ejecutando el método `EsIgual` (imprima en pantalla un mensaje indicando si, efectivamente, son iguales)