Trabajo Práctico 2

Tipos Abstractos y Estructuras de Datos

Introducción a la Computación

1er cuatrimestre 2017

Observaciones generales:

- El trabajo se debe realizar en grupos de tres personas.
- El código fuente debe estar bien comentado.
- El informe y el código fuente deben enviarse por mail a la lista de docentes de la materia: `icm-doc@dc.uba.ar`.
- El programa entregado debe correr correctamente con el `Python 2.7.x` instalado en las computadoras del laboratorio Turing.
- Se evaluará la correctitud, claridad y modularidad del código y el informe entregado.
- La fecha límite de entrega es el jueves 22 de junio a las 23:59.

La empresa ME LA JUEGO S.A. ha decidido contratarlos para que formen parte del equipo de desarrollo de su ambicioso proyecto SCRABBLE ONLINE. Tal como su nombre lo indica se trata de una versión para Internet del famoso juego de palabras[1].

Cada uno de los jugadores de una sesión de SCRABBLE verá en la pantalla de su computadora (o dispositivo móvil) la imagen del tablero y de las fichas que posee para formar palabras. Una vez que un jugador coloca una palabra sobre el tablero el juego debe (entre otras cosas) chequear que la palabra sea válida. El trabajo que se les encomienda es relativo a este chequeo.

El juego debe poder jugarse en distintos idiomas. Por ahora ya está garantizada su comercialización en Francia, Argentina, Brasil y Alemania. De todas formas, como la intención es poder venderlo también en otros mercados, se espera que el software diseñado tenga buenos tiempos de respuesta independientemente del idioma en el que se esté jugando. Más precisamente, los tiempos de corrida deben ser independientes de la cantidad de palabras que posea el idioma seleccionado. Ésto es muy importantes: el tiempo que le demande al programa comprobar si una palabra es válida será una gran parte del tiempo que los jugadores estarán esperando frente a su pantalla hasta que el juego pueda proseguir. En consecuencia se les requiere implementar el tipo de dato que se describe a continuación respetando los órdenes de complejidad temporal indicados:

TAD LÉXICO

- *NuevoLéxico()* $\rightarrow$ Léxico: Devuelve un nuevo léxico vacío en $O(1)$.

[1] Para más información sobre el Scrabble visitar `http://es.wikipedia.org/wiki/Scrabble`

- *L.CantidadDePalabras()* $\rightarrow \mathbb{Z}$: Devuelve la cantidad de palabras del léxico *L* en $O(1)$.
- *L.AgregarPalabra(P)*: Agrega la palabra *P* al léxico *L* en *O(|P|)*. Si *P* ya existe en *L* la operación no tiene efecto sobre *L*
- *L.EliminarPalabra(P)*: Borra a la palabra *P* del léxico *L* en *O(|P|)*. Si *P* no existe en *L* la operación no tiene efecto sobre *L*
- *L.Existe(P)* $\rightarrow \mathbb{B}$: Devuelve TRUE si la palabra *P* forma parte del léxico *L* y FALSE en caso contrario en *O(|P|)*.

Se pueden hacer las siguientes suposiciones:

- las palabras sólo se ingresarán en minúscula;
- ninguna palabra tendrá tildes, eñes, ni diéresis (“ü”).

De ser necesario, se pueden definir TADs auxiliares. La entrega del trabajo práctico debe incluir:

1. un informe que contenga:
   a) las estructuras de representación propuestas para los TADs que se hayan utilizado;
   b) los invariantes de representación de las estructuras propuestas (en español);
   c) los algoritmos en pseudocódigo, justificando en cada caso que cumplen con los órdenes requeridos;
2. la implementación del TAD Léxico en Python.

El TAD Léxico debe implementarse en una clase llamada `Lexico`. Su código fuente debe estar en un archivo llamado `TIPO_LEXICO.py` y respetar la siguiente signatura:

```
class Lexico:
  def __init__(self)
  def cantidadDePalabras(self)
  def agregarPalabra(self, p) # p:string, correspondiente a una palabra
  def eliminarPalabra(self, p) # p:string, correspondiente a una palabra
  def existe(self, p) # p:string, correspondiente a una palabra
```