El Ahorcado Multijugador

El trabajo práctico 2, es una extensión del Trabajo Práctico 1, que incorpora nuevas funcionalidades que involucran manejo de archivos e introduce algunas modificaciones a las reglas del juego.

Objetivo

Escribir un programa en Python, que ofrezca a un grupo de N personas, jugar al ahorcado, respetando las reglas y condiciones que se dan a continuación.

Diccionario de Palabras

Al iniciar el programa y antes de comenzar a jugar, se debe generar un archivo de texto, llamado palabras.txt. Dicho archivo, contendrá las palabras que podrán ser otorgadas a los jugadores.

Para generar las palabras, se deberán procesar 3 textos diferentes, que serán provistos. Los textos pueden ser leídos de a lo sumo una línea a la vez, no pudiendo estar cargados en su totalidad en memoria. Los textos, deben ser procesados, de forma tal de obtener sólo palabras válidas, formadas sólo por letras; y no deben estar repetidas. El archivo palabras.txt debe estar ordenado alfabéticamente.

Se proveerá también de un archivo delimitado por comas, llamado reemplazar.csv, que contendrá como primer dato el caracter o la cadena a reemplazar, y como segundo dato, la cadena o caracter por la cual reemplazar. Por ejemplo:

é,e

IX),

La primer línea está indicando que todos los caracteres é, deben ser reemplazados por e.

La segunda línea está indicando que la cadena IX) sea reemplazado por un caracter nulo ó por la secuencia de caracteres en blanco que hay entre la “,” y el fin de línea.

Deberán agregar al archivo todos aquellos caracteres que sean necesarios reemplazar, para que al momento de realizar el proceso, asegure que las palabras sólo contengan letras.

Para la obtención del archivo palabras.txt, se debe generar una serie de procesos que:

1) En base a cada texto, depure el mismo, y obtenga un archivo **palabras_texto_N.txt**, ordenado alfabéticamente
2) Realice un proceso de mezcla, de los 3 archivos **palabras_texto_N**.txt, y obtenga el archivo **palabras.txt**

Al finalizar el proceso, deberán informar por pantalla, por cada longitud de palabra, cuantas palabras hay, y el total de palabras formadas.

Desarrollo del Juego

Para iniciar el juego, el programa debe preguntar la cantidad de personas que jugarán, que no deben ser más de N.

A continuación solicitará el ingreso de los nombres de cada una de las personas.

Luego debe otorgar aleatoriamente, el orden en que jugarán los participantes, e informar el mismo.

A continuación, el programa debe solicitar la longitud de la palabra a adivinar, que no podrá ser menor a L caracteres. Si no hay palabras con la longitud ingresada, solicitar un nuevo número.

Cada participante tendrá que adivinar una palabra que debe ser otorgada aleatoriamente por el programa, entre todas las posibles que cumplan con la longitud ingresada.

Cada jugador, tendrá un máximo de M desaciertos por palabra.

En cada turno, el jugador deberá ingresar una letra. Si la letra se encuentra en la palabra, se mostrará en las posiciones donde se encuentra, y el jugador sumará X puntos por haber acertado. Si no acierta, se le restan Y puntos. En cualquiera de los casos, el turno pasa al siguiente jugador.

El primer jugador que acierte la palabra, es quien gana la partida; y suma Z puntos por haber adivinado la palabra.

Si ninguno la adivina, gana el programa.

Cuando un jugador llega al máximo M de desaciertos, la partida continúa con el resto de los jugadores.

En cada Turno, se debe mostrar el nombre de quien está participando, cuales son sus aciertos y desaciertos hasta el momento, y el puntaje que tiene hasta el momento.

Al finalizar cada partida, mostrar el resultado de la partida, indicando para cada jugador la palabra que debía adivinar, cantidad de aciertos y desaciertos, y el puntaje obtenido; y luego preguntar si desean jugar otra partida.

Si se juega más de una partida, acumular los resultados de las partidas y mostrar los Resultados Generales, indicando la cantidad de partidas jugadas; mostrando por jugador, los datos ordenados por Puntaje Total, junto con la cantidad de aciertos y desaciertos.

En cada nueva partida, volver a solicitar la longitud de la palabra a adivinar.

A partir de la segunda partida, el orden de los jugadores, deberá generarse nuevamente, pero esta vez, deberá tener en cuenta quien ganó la partida anterior, y colocarlo en primer lugar; para el resto de los participantes, se tendrá en cuenta el puntaje acumulado, colocando primero a los de mayor puntaje, y a igual puntaje, decidir aleatoriamente entre estos.

Como resultado de la finalización del juego, debe generarse un archivo de texto delimitado por comas, llamado partida.csv, que contenga:

nombre del jugador, total de aciertos, total de desaciertos, puntaje total, palabras

El archivo debe estar ordenado por puntaje total. Las palabras deben estar separadas por blancos.

Los valores: L, N, M, X, Y, Z, deben ser leídos de un archivo, llamado configuracion.txt, que podría tener el siguiente formato:

MAX_USUARIOS 10

LONG_PALABRA_MIN 5

MAX_DESACIERTOS 7

PUNTOS_ACIERTOS 2

PUNTOS_DESACIERTOS 1

PUNTOS_ADIVINA 30

Si por algún motivo, uno o más valores no pueden ser recuperados del archivo de configuración, deben ser establecidos con valores por defecto. Una vez establecido los valores, mostrar una pantalla con los mismos, indicando como fue asignado el valor.

Validación de Datos

El ingreso de los datos debe ser validado respetando según corresponda, el tipo de dato, los rangos, o el formato que debe respetar el texto ingresado. Los nombre de los jugadores sólo podrán contener letras y espacios en blancos. Se recomienda todo texto, convertirlo a mayúsculas.

Presentación

El programa debe respetar las reglas de la programación estructurada, evitando especialmente, la redundancia de código.

Cada grupo deberá subir al campus:

1. Documento que gráficamente muestre la solución modular implementada.
   Descripción de las estructuras de datos que utilizarán, su función y su interacción. Puede ser mediante un gráfico.
   Se recomienda aplicar las técnicas mencionadas en clase, respetando la etapa de diseño, llegando a una solución modular adecuada, antes de comenzar a codificar.
   Se debe respetar el estilo de programación dado en clase, tomando como referencia el PEP 8.
2. El programa fuente de la aplicación. Cada función debe tener su correspondiente comentario compuesto por Autor: indicando el nombre del alumno que lo codificó, y una breve descripción de lo que realiza.