Universidad Rey Juan Carlos

Normas del examen

- En todos los ejercicios se valorará la eficiencia de las estructuras de datos utilizadas y del código implementado.
- Si un código resuelve el problema de forma no eficiente se podrá considerar como no válido, aunque la salida sea correcta.
- Para comenzar a resolver el examen, descarga el fichero con el código de apoyo del Aula Virtual, descomprime ese fichero y copia el contenido de la carpeta src dentro de la carpeta src de tu proyecto.
- La entrega se basará en un único fichero ZIP que contendrá la carpeta src con todo el código implementado.
- Puedes implementar tantas clases y métodos como necesites, pero sin modificar las cabeceras de los métodos propuestos.
- No se permite la utilización de ningún tipo de apoyo, ni prácticas realizadas previamente. Cualquier intento de utilización de este tipo de herramientas implicará el suspenso automático de la convocatoria completa y las medidas disciplinarias oportunas.

Ejercicio 1 [3 puntos]

Se pide implementar un método que, dado un árbol genérico T y dos de sus nodos $n1$ y $n2$, cree una copia del árbol cuya raíz es el ancestro común de $n1$ y $n2$ de mayor profundidad. Dicha funcionalidad deberá estar incluida en el método `createCommonTreeAncestor`, cuya cabecera está definida en la clase `CommonAncestorTree`. Se proporcionan test unitarios para verificar su correcto funcionamiento.

Ejercicio 3 [5 puntos]

La empresa QuienEsQuien S.L. nos ha pedido implementar su nueva aplicación destinada a desarrollar una Inteligencia Artificial que sea capaz de jugar al quién es quién. Dicha aplicación debe ser capaz de cargar tanto los juegos como las respuestas de los usuarios a las preguntas desde un fichero para poder analizarlo. Los juegos se componen de preguntas que siempre se responden con Sí/No, y al terminar de responder a las preguntas, se le indicará al usuario el personaje que ha elegido. Dependiendo de la respuesta, se elegirá la siguiente pregunta, y así hasta llegar al final del juego. En particular, la primera pregunta será común a todos los usuarios y, a continuación, se irán eligiendo el resto de las preguntas dependiendo de si el usuario responde afirmativa o negativamente. En concreto, se pide implementar la siguiente funcionalidad:

a) **[0.5 puntos]** Definir la estructura de datos necesaria para almacenar la información sobre los juegos. Se debe guardar información sobre un único juego que se sobrescribirá en caso de cargar un nuevo juego. Implementa

los métodos `loadTest`, `resetIterator`, `hasMoreQuestions`, y `nextQuestion`, descritos en la clase `GuessWho`.

b) **[1.5 puntos]** Implementar el método `longestGame`, que nos indica cuál es el mayor número de preguntas que nos puede realizar la IA hasta llegar a una respuesta en el juego que esté cargado (no se debe contar la respuesta).

c) **[0.5 punto]** Implementar el método `countResponses`, que nos indicará cuántos personajes tiene cargado el juego actual.

d) **[1 punto]** Implementar el método `playGame`, que recibe un fichero con la jugada y enumera las preguntas que se van haciendo hasta llegar al personaje, o directamente da el personaje

e) **[1.5 puntos]** Implementar el método `showTree`, que devuelve un `String` con la representación gráfica del juego. Comenzando desde la primera pregunta, se mostrarán las preguntas cada una en una línea, con tabuladores al inicio en función del orden de la pregunta (revisa el test unitario para comprobar cómo debe mostrarse).