Instituto Tecnológico de Costa Rica
Área Ingeniería en Computadores
Lenguajes, Compiladores e intérpretes (CE3104)
Primer Semestre 2020
Tarea# 1 Grupo1

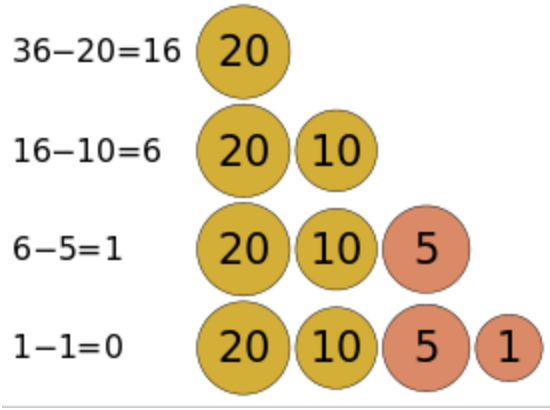
1. **Objetivo General**
   - ➔ Desarrollar una aplicación que permita reafirmar el conocimiento del **paradigma de programación funcional**.

2. **Objetivos Específicos**
   - ➔ Crear una aplicación que resuelva el problema utilizando Racket.
   - ➔ Aplicar los conceptos de programación funcional.
   - ➔ Crear y manipular listas como estructuras de datos.

3. **Datos Generales**
   - ➔ El valor de la Tarea: 7,5%
   - ➔ Nombre código: TicTacToe
   - ➔ La tarea debe ser implementada en grupos de no más de 3 personas.
   - ➔ La **fecha de entrega** es: **1 de Julio 2020** hasta las 11:59pm.
   - ➔ Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo al reglamento.
   - ➔ El correo de entrega debe tener como Asunto: CE-3104 IS2020 TC1 Gr01 debe ser enviado a mriveramTec@gmail.com. Y se debe copiar a todos los integrantes del grupo y al asistente del curso Kevin (kevin.gonzalezs0795@gmail.com).

4. **Algoritmos golosos (Greedy Algorithms)** (Wikipedia, 2020)**.**

   4.1. Un algoritmo codicioso es un algorítmico que sigue la heurística de resolución de problemas de hacer la elección localmente óptima en cada etapa con la esperanza de encontrar un óptimo global. En muchos problemas, una estrategia codiciosa generalmente no produce una solución óptima, pero una heurística codiciosa puede ofrecer soluciones locales óptimas que se aproximan a una solución óptima global en un tiempo razonable. El siguiente ejemplo muestra como un algoritmo codicioso entregaría el vuelto de 36 colones a un cliente teniendo únicamente monedas de 1, 5, 10 y 20.



4.2. Cinco componentes de un algoritmo codicioso:

4.2.1. Conjunto de candidatos a partir del cual se crea una solución.

4.2.2. Función de selección, que elige el mejor candidato para agregar a la solución.

4.2.3. Función de viabilidad, que se usa para determinar si un candidato puede usarse para contribuir a una solución.

4.2.4. Función objetivo, que asigna un valor a una solución, o una solución parcial.

4.2.5. Función de solución, que indicará cuándo hemos descubierto una solución completa.

5. **Descripción del caso.**

Tic Tac Toe (gato-tres en línea): es un juego de lápiz y papel entre dos jugadores: O y X, que marcan los espacios de un tablero de 3×3 alternadamente. Un jugador gana si consigue tener una línea de tres de sus símbolos: la línea puede ser horizontal, vertical o diagonal.

El juego se compone de:

5.1. **Interfaz Gráfica.**

5.1.1. Debe existir un tablero de **MXN** (donde M y N son especificados por el usuario al iniciar el juego, el mínimo del tablero es 3 X 3 y el máximo de 10 X 10).

5.1.2. El juego es contra la máquina.

5.1.3. El sistema debe mostrar una interfaz gráfica amigable con el usuario.

5.1.4. El usuario siempre utilizara la ficha X y el sistema la ficha O.

5.1.5. El usuario seleccionara por medio del mouse la posición en la que desea poner su ficha (X).

5.2. **Reglas del Juego.**

5.2.1. Para iniciar el juego se realizará de la siguiente manera.

5.2.2. > (TTT 3 4)

5.2.2.1. Donde 3 se refiere a la cantidad de columnas y 4 a la cantidad de filas del tablero.

5.2.3. Se debe mostrar la interfaz.

5.2.4. Solicitar la posición donde se desea poner su ficha.

5.2.5. Enseguida el sistema activa su algoritmo voraz.

5.2.6. Marca la mejor posición y se repiten los pasos 5.2.4 al 5.2.6 hasta que exista un ganador.

5.2.6.1. Un ganador existe cuando al menos hay 3 símbolos en línea y se llegue al final del tablero de lado a lado.

6. **Entregables**

6.1. Código fuente comentado.

6.1.1. La interfaz gráfica debe ser entregada en un archivo.

6.1.2. La lógica del juego debe ser entregada en un archivo independiente de la interfaz.

6.2. **Manual de usuario.**

7. **Documentación**

1. Se deberá entregar un documento que contenga:
   - **1.1. Descripción detallada de los algoritmos/arquitectura desarrollada.**
   - **1.2. Descripción de las todas las funciones implementadas.**
   - **1.3. Descripción de la ejemplificación de las estructuras de datos desarrolladas.**
   - 1.4. Problemas sin solución: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.
   - 1.5. Plan de Actividades realizadas por estudiante: Este es un planeamiento de las actividades que se realizaran para completar la tarea, este debe incluir descripción de la tarea, tiempo estimado de completitud, responsable a cargo y fecha de entrega.
   - 1.6. Problemas encontrados: descripción detallada, intentos de solución sin éxito, soluciones encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.
   - 1.7. Conclusiones y Recomendaciones del proyecto.
   - 1.8. Bibliografía consultada en todo el proyecto
2. Bitácora en digital, donde se describen las actividades realizadas, desde reuniones con el compañero de trabajo, investigaciones, consultas, etc. Está se puede encontrar hecha a mano, se debe describir todo por más insignificante que sea, esto demostrará si ustedes están trabajando en realidad. Este es su diario de trabajo, llevan seguimiento de todo en el tiempo, imaginen que, si un compañero los releva en su trabajo, le bastaría con leer sus bitácoras para seguir el trabajo.

8. **Evaluación**

1. El proyecto tendrá un valor de un 70% de la nota final, debe estar funcional.
2. La documentación tendrá un valor de un 20% de la nota final, cumplir con los requerimientos especificados en la documentación no significa que se tienen todos los puntos, se evaluará que la documentación sea coherente, acorde al tamaño del proyecto y el trabajo realizado, no escatimen en documentación.
3. La defensa tendrá un valor de 10%, todos los integrantes del grupo deben participar, se debe preparar un par de filminas con la explicación de la arquitectura y funciones más significativas implementadas.
4. Cada grupo recibirá una nota en cada uno de los siguientes apartados Código, Documentación y defensa.
5. El profesor no sólo evaluará la funcionalidad del proyecto, esto quiere decir que, aunque el

proyecto este 100% funcional esto no implica una nota de un 100, ya que se evaluarán aspectos de calidad de código, aplicación del **paradigma funcional (no se permite el uso de let, map, apply, set)**, calidad de documentación interna y externa y trabajo en equipo.

6. No se revisarán funcionalidades parciales, ni funcionalidades no integradas.
7. Es responsabilidad de cada miembro del grupo conocer su código, el profesor puede preguntar a cualquier miembro del grupo que le explique alguna funcionalidad/porción de código.
8. De las notas mencionadas en el punto 4 se calculará la Nota Final del Proyecto.
9. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
10. Aun cuando el código, la documentación y la defensa tienen sus notas por separado, se aplican las siguientes restricciones
    - 10.1. Si no se entrega documentación, automáticamente se obtiene una nota de 0.
    - 10.2. Si no se entrega el punto 1 de la documentación se obtiene una nota de 0.
    - 10.3. Si el código y la documentación no se entregan en la fecha indicada se obtiene una nota de 0.
    - 10.4. Si el código no compila se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
    - 10.5. Si el grupo no cuenta con los equipos necesarios para realizar la revisión y no avisó al profesor de esta situación obtendrá una nota de 0.
    - 10.6. El código debe ser desarrollado en **Racket** utilizando el **paradigma de programación funcional**, en caso contrario se obtendrá una nota de 0.
    - 10.7. **NO** presentarse a la defensa se obtendrá una nota de 0.
11. Cada grupo tendrá como máximo 30 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cual se recomienda tener todo listo antes de ingresar a la defensa esto implica tener casos de prueba listos.
12. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
13. Cada grupo es responsable de llevar los equipos requeridos para la revisión.
14. Durante la revisión únicamente podrán participar los miembros del grupo, asistentes, otros profesores y el coordinador del área.
15. Las revisiones se realizan con los estudiantes matriculados en el curso, cualquier persona fuera de estos y los mencionados en el punto 13, no pueden participar en la revisión.
16. Después de enviada la nota final del proyecto el estudiante tendrá un máximo de 3 días hábiles para presentar un reclamo siempre y cuando la funcionalidad esté completa.

9. **Referencias**

Guzman, J. E. (2006). *Introducción a la programació con Scheme.* Cartago: Editorial Tecnológica de Costa Rica.

Wikipedia. (2020, 3 30). *Greedy algorithm*. Retrieved from Wikipedia: https://en.wikipedia.org/wiki/Greedy_algorithm