Instituto Tecnológico de Costa Rica
Escuela Ingeniería en Computadores
Paradigmas de Programación
Segundo Semestre 2025
Tarea# 2

1. **Objetivo General**

   ➔ Desarrollar una aplicación que permita reafirmar el conocimiento del **paradigma de programación funcional, recursividad y estructuras de datos**.

2. **Objetivos Específicos**

   ➔ Crear una aplicación que resuelva el caso utilizando Racket.

   ➔ Aplicar los conceptos de programación funcional.

   ➔ Crear y manipular listas como estructuras de datos.

3. **Datos Generales**

   ➔ El valor de la Tarea: 12.5%

   ➔ Nombre código: BusCEMinas

   ➔ La tarea debe ser implementada en grupos de no más de 3 personas.

   ➔ La **fecha de entrega** es 27/Setiembre/2025.

   ➔ Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo al reglamento.

4. **Descripción del caso.**

   Buscaminas (nombre original en inglés: Minesweeper) es un género de videojuegos de lógica que generalmente se juega en computadoras personales. El videojuego presenta una cuadrícula de casillas en las que se puede hacer clic, donde hay «minas» ocultas (representadas como minas navales en el videojuego original) esparcidas por todo el tablero. El objetivo es limpiar el tablero sin detonar ninguna mina, con la ayuda de pistas sobre el número de minas vecinas en las casillas circundantes (Wikipedia, 2025).

   4.1. **Requerimientos Funcionales**

   - Generar automáticamente un tablero de tamaño configurable (ej. 8x10, 10x12, etc.) con un número aleatorio de minas.
     ➔ (BuscaCE 8 11 [Nivel Fácil 10%- Medio 15% - Difícil 20%])
     ➔ Mínimo 8 – Máximo 15.
   - Permitir al usuario descubrir una celda:
     - Si la celda es una mina → el juego termina (derrota).
     - Si no es mina → mostrar el número de minas adyacentes.

- Si es un “0” (ninguna mina cerca) → descubrir recursivamente las celdas vecinas.
- Permitir al usuario marcar una celda con una bandera.
- Detectar cuando el jugador gana (todas las celdas sin mina descubiertas).

4.2. **Requerimiento NO Funcionales.**

4.2.1. Se require al menos 2 archivos, uno para la interfaz gráfica y otro para la lógica del juego.

4.3. **Interfaz Gráfica.**

4.3.1. Se debe mostrar una interfaz que permita visualizar el tablero.

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

4.3.3. Se debe seleccionar una casilla del tablero realizando un click.

5. **Entregables**

5.1. Código fuente comentado.

5.2. Manual de usuario.

6. **Documentación Técnica**

1. Se deberá entregar un documento que contenga:

1.1. Descripción de las funciones implementadas.

1.2. Ejemplificación de las estructuras de datos desarrolladas.

i **(((0 0) (0 3))**
(1 3) (0 1)))

Esta estructura representa el tablero en este caso un tablero de 2x2. Cada sublista es una casilla del tablero donde la primera posición un 0 indica que NO es una Mina y la segunda posición las minas adyacentes.

(MinaSi/No NumeroMinasAdyacentes).

Aquí van las demás estructuras…

1.3. Descripción detallada de los algoritmos desarrollados (Incluir diagrama).

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 realizarán para completar la tarea, este debe incluir descripción de la tarea, tiempo estimado de completitud, fecha de entrega y responsable a cargo.

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.

7. **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.
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 variables -let, map, apply-), calidad de documentación interna y externa y trabajo en equipo (plan de actividades).
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 3 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.
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 14, 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.

8. **Referencias**

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

Racket. (2017, 08 15). *The Racket Graphical Interface Toolkit*. Retrieved from Racket: http://download.racket-lang.org/releases/6.9/doc/gui/

Wikipedia. (2025, September 04). Buscaminas. Retrieved from Wikipedia: https://es.wikipedia.org/wiki/Buscaminas