Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Área de Ingeniería en Computadores
Lenguajes, Compiladores e intérpretes (CE3104)
Primer Semestre 2023
Tarea# 1

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 del proyecto: 7,5%
   - ➔ Nombre código: PDC
   - ➔ La tarea debe ser implementada en grupos de no más de 3 personas.
   - ➔ La **fecha de entrega** es 19/Mayo/2023
   - ➔ Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo al reglamento.

4. **Descripción del caso.**

El problema del caballo (Wikipedia, Problema del caballo, 2023) es un antiguo problema matemático en el que se pide que, teniendo una cuadrícula de n x n casillas y un caballo de ajedrez colocado en una posición cualquiera (x, y), el caballo pase por todas las casillas (Wikipedia, Caballo (ajedrez), 2023) una sola vez. Lo que resulta en n^2-1 movimientos.

Se deben programar tres funciones principales:

- PDC-Sol esta función recibe dos argumentos un entero que indica el tamaño de la matriz y un par ordenado que indica donde se encuentra inicialmente el caballo.
  >(PDC-Sol 8 '(1 1))
  (solución)
  Como resultado debe retornar una estructura (a definir por el estudiante) con una solución de la secuencia de casillas que el caballo va visitando.
- PDC-Todas al igual que la función anterior recibe dos parámetros, un entero que indica el tamaño de la matriz y un par ordenado que indica donde se encuentra inicialmente el

caballo y retorna todas las soluciones posibles.
>(PDC-Todas 8 '(1 1))
((solucion1)
(solucion2)
…
(solucionN))

- PDC-Test al igual que las funciones anteriores recibe dos argumentos un entero que indica el tamaño de la matriz y una estructura solución, la idea es probar la solución identificada y representarla en una matriz.

  >(PDC-Test 8 '(solucion))
  ((01 17 00 00 09 00 00 00)
  (00 00 10 18 00 00 00 00)
  (00 02 00 00 00 08 00 00)
  (00 00 16 11 00 00 00 07)
  (00 00 03 00 00 00 00 00)
  (00 00 12 15 00 00 06 00)
  (00 00 00 04 00 14 00 00)
  (00 00 00 13 00 05 00 00)
  pts)

- PDC-Pintar al igual que la función anterior recibe dos argumentos un entero que indica el tamaño de la matriz y una estructura solución, la idea es la misma probar la solución encontrada pero utilizando la librería Racket Graphical Interface Toolkit (Racket, The Racket Graphical Interface Toolkit, 2017) y representarla de una forma gráfica/animada a definir por él estudiante.

  >(PDC-Paint 8 '(solucion))

5. **Entregables**
   5.1. Código fuente comentado.
      5.1.1. La interfaz gráfica debe ser entregada en un archivo.
      5.1.2. La lógica del juego debe ser entregada en un archivo independiente de la interfaz.
   5.2. Manual de usuario.

6. **Documentación**
   1. Se deberá entregar un documento que contenga:
      1.1. Descripción detallada del (los) algoritmo(s) de solución desarrollado(s) (con su respectivo(s) diagrama(s)).
      1.2. Descripción de las funciones implementadas.
      1.3. Descripción de la ejemplificación/uso de las estructuras de datos desarrolladas.

((**1 0 0 2 0 0 0 3)** (1 0 0 0 0 0 0 0) (1 0 0 0 0 0 0 0) (1 0 0 0 0 0 0 0) (2 0 0 0 0 0 0 0) (2 0 0 0 0 0 0 0) (1 0 0 0 0 0 0 0) (2 0 0 0 0 0 0 0))

(1 0 0 2 0 0 0 3) se refiere a la primera fila del tablero donde 1 es el valor de la primera casilla que visito el caballo, 2 significa la segunda posición que visito el caballo.

1.4. Problemas sin solución: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.

1.5. Problemas encontrados: descripción detallada, intentos de solución sin éxito, soluciones encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.

1.6. 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.

1.7. Conclusiones.

1.8. Recomendaciones.

1.9. 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 código 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. Preparar una pequeña presentación 2-3 diapositivas si fuera necesario con la explicación del algoritmo de solución y enfocado en los ítems de la rúbrica.
    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 ni cualquier otra primitiva que no sea del paradigma funcional)**, 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 1.2 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 penalizara 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). *racket-lang.org*. Retrieved from Racket: https://racket-lang.org/

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

Wikipedia. (2023, 05 05). *Caballo (ajedrez)*. Retrieved from Wikipedia: https://es.wikipedia.org/wiki/Caballo_(ajedrez)

Wikipedia. (2023, 05 05). *Problema del caballo*. Retrieved from Wikipedia: https://es.wikipedia.org/wiki/Problema_del_caballo#El_problema_del_caballo_en_la_literatura