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: Bla**CE**Jack
   - ➔ La tarea debe ser implementada en grupos de no más de 3 personas.
   - ➔ La **fecha de entrega** es: **19 de Marzo 2021**.
   - ➔ Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo al reglamento.
   - ➔ La entrega se realizará por la plataforma TEC Digital en su sección correspondiente.

4. **Descripción del caso**
   Black Jack o veintiuno, es un juego de cartas, propio de los casinos con una o más barajas inglesas de 52 cartas sin los comodines, que consiste en sumar un valor lo más próximo a 21 pero sin pasarse. En un casino cada jugador de la mesa juega únicamente contra el crupier, intentando conseguir una mejor jugada que este. El crupier está sujeto a reglas fijas que le impiden tomar decisiones sobre el juego. Por ejemplo, está obligado a pedir carta siempre que su puntuación sume 16 o menos, y obligado a plantarse si suma 17 o más. Las cartas numéricas suman su valor, las figuras suman 10 y el As vale 11 o 1, a elección del jugador. En el caso del crupier, los Ases valen 11 mientras no se pase de 21, y 1 en caso contrario. La mejor jugada es conseguir 21 con solo dos cartas, esto es con un As más carta de valor 10. Esta jugada se conoce como Blackjack o 21 natural. Un Blackjack gana sobre un 21 conseguido con más de dos cartas.
   El Juego se compone de:

   4.1. **Reglas del Juego**
      4.1.1. Para iniciar el juego se realizará de la siguiente manera.

4.1.1.1. > (bCEj X)

Donde X es una lista con los nombres de los jugadores de esa partida, la cantidad máxima de jugadores es 3. El juego será entre el computador (crupier) y la cantidad de jugadores especificada.

4.1.2. Las reglas del juego son las descritas en la sección **Descripción del caso** únicamente esas NO existirán, separar, asegurar o rendirse.

4.1.3. Una vez iniciado el juego, el jugador uno podrá pedir una carta más o plantarse ahí. En caso de que se plante a ese jugador no se le cederá más el turno, así mismo se procederá con los siguientes jugadores. Hasta que todos decidan no solicitar más cartas o plantarse. En ese momento el crupier muestra su carta oculta y empieza a solicitar cartas (El crupier solo solicitara cartas cuando su puntuación sea 16 o menos en caso contrario se plantara).

4.1.4. Al final del juego se mostrará la tabla de posiciones con los jugadores y su puntuación.

4.2. **Interfaz gráfica.**

4.2.1. Se debe mostrar una interfaz que permita visualizar el maso y las cartas de los jugadores excepto la primera.

4.2.2. Debe existir una opción (botón/instrucción) para que los jugadores soliciten una nueva carta o se planten.

4.2.3. El juego es contra la máquina (quien es el Crupier).

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

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 de las estructuras de datos desarrolladas.**
     ((1 (5 4)) (2 (1 10)) (3 (10 11))) Esta es la lista de jugadores con sus cartas donde (1 (5 4)) se refiere al primer jugador y posee las cartas 5 y 4 la carta 5 siempre será la carta oculta.
   - 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 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. Preparar una pequeña presentación 2-3 diapositivas con la explicación del algoritmo de solución.
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 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.

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

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