Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación
Área de Ingeniería en Computadores
Lenguajes, Compiladores e intérpretes (CE3104)
Primer Semestre 2019
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 DrRacket.
   - ➔ 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: Wazitico
   - ➔ La tarea debe ser implementada en grupos de 3 personas.
   - ➔ La **fecha de entrega** es 20/Agosto/2019.
   - ➔ Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo al reglamento.

4. **Grafos**
   - 4.1. Un grafo es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.
   - 4.2. **Grafo Dirigido**, es un tipo de grafo en el cual las aristas tienen un sentido definido, a diferencia del grafo no dirigido, en el cual las aristas son relaciones simétricas y no apuntan en ningún sentido.
   - 4.3. **Grafo Mixto,** es aquel que se define con la capacidad de poder contener aristas dirigidas y no dirigidas. Tanto los grafos dirigidos como los no dirigidos son casos particulares de este.

5. **Descripción del caso.**

   La presente tarea tiene como objetivo la implementación de un grafo mixto que simule la famosa aplicación Waze.

   Waze es una aplicación social de tránsito automotor en tiempo real y navegación asistida por GPS desarrollada por Waze Mobile. El 11 de junio 2013, Google completó la adquisición de Waze en $966 millones de dólares.

5.1. **Funcionalidades.**

5.1.1. Construir mapa: El sistema debe proveer la carga de una ciudad (Grafo) definida por lugares (nodos) y carreteras (aristas) de una sola vía o vía en ambos lados como en la vida real. Las carreteras tendrá un peso, que indicara distancia entre un lugar y otro.

5.1.2. Definición origen: El sistema debe proveer la definición de un lugar como el punto de partida de donde se parte la búsqueda de opciones de rutas hacia el destino.

5.1.3. Definición destino: El sistema debe proveer la definición de un lugar como el destino final.

5.1.4. Búsqueda: El sistema debe realizar la búsqueda de todas las rutas posibles para llegar del origen al destino.

5.2. **Interfaz Gráfica.**

5.2.1. El sistema debe proveer una vía amigable de solicitar el mapa de la ciudad, el origen y el destino.

5.2.2. El sistema debe graficar el mapa.

5.2.3. El sistema debe permitir mostrar todas las rutas, pero debe mostrarse por defecto la más corta.

5.2.4. Al seleccionar una ruta se debe mostrar la distancia del origen al destino.

6. **Entregables**

6.1. Código fuente comentado.

6.2. Manual de usuario.

6.3. **Documentación Técnica**

1. Se deberá entregar un documento que contenga:
   - **1.1. Descripción de las funciones implementadas.**
   - **1.2. Descripción de las estructuras de datos desarrolladas.**
   - **1.3. Descripción detallada de los algoritmos desarrollados.**
   - 1.4. Problemas conocidos: 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: Esta tarea debe realizarse antes de empezar con el proyecto y lo que busca es que se realice un plan, este debe contener al menos: la lista de actividades, responsable y fecha de entrega de la actividad.
   - 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**
   7.1. El proyecto tendrá un valor de un 70% de la nota final, debe estar funcional.
   7.2. 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.
   7.3. La defensa tiene un valor de 10%, todos los miembros deben participar y todos deben conocer el código.
   7.4. Cada grupo recibirá una nota en cada uno de los siguientes apartados Código, Documentación y defensa.
   7.5. 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**, calidad de documentación interna y externa y trabajo en equipo.
   7.6. No se revisarán funcionalidades parciales, ni funcionalidades no integradas.
   7.7. 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.
   7.8. De las notas mencionadas en el punto 3 se calculará la Nota Final del Proyecto.
   7.9. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
   7.10. Aun cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
      7.10.1. Si no se entrega documentación, automáticamente se obtiene una nota de 0.
      7.10.2. Si no se entrega el punto 4 de la documentación se obtiene una nota de 0.
      7.10.3. Si el grupo no se presenta a la defensa tendrá una nota de 0.
      7.10.4. Si el código y la documentación no se entregan en la fecha indicada se obtiene una nota de 0.
      7.10.5. Si el código no compila se obtendrá una nota de 0, por lo cual se recomienda realizar la defensa con un código funcional.
      7.10.6. 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.
      7.10.7. El código debe ser desarrollado en **DrRacket** utilizando el **paradigma de programación funcional**, en caso contrario se obtendrá una nota de 0.
   7.11. 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.
   7.12. 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.
   7.13. Cada grupo es responsable de llevar los equipos requeridos para la revisión.
   7.14. Durante la revisión únicamente podrán participar los miembros del grupo, asistentes, otros profesores y el coordinador del área.
   7.15. 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.
   7.16. 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 programació 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/