Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Área de Ingeniería en Computadores
Lenguajes, Compiladores e intérpretes (CE3104)
Primer Semestre 2019
Tarea# 2

1. **Objetivo General**
   - ➔ Desarrollar una aplicación que permita reafirmar el conocimiento del **paradigma de programación lógico y orientado a objetos**.

2. **Objetivos Específicos**
   - ➔ Crear una aplicación que se comporte como un Doctor Experto utilizando Prolog.
   - ➔ Aplicar los conceptos de programación lógico.
   - ➔ Crear y manipular listas como estructuras de datos.

3. **Datos Generales**
   - ➔ El valor del proyecto: 12,5%
   - ➔ Nombre código: DrWazeLog
   - ➔ La tarea debe ser implementada en grupos de no más de 3 personas.
   - ➔ La **fecha de entrega** es 28/Mayo/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.**

Los sistemas expertos (SE) son aplicaciones de cómputo que involucran experiencia no algorítmica, para resolver cierto tipo de problema. Por ejemplo, los sistemas expertos se usan para el diagnóstico al servicio de humanos y máquinas. Existen SE que juegan ajedrez, que plantean decisiones

financieras, que configuran computadoras, que supervisan sistemas de tiempo real, que deciden políticas de seguros, y llevan a cabo demás tareas que requieren de experiencia humana.

La presente tarea tiene como objetivo desarrollar un SE para un sistema de ubicación y guía del tránsito, la interfaz debe ser completamente natural utilizando el lenguaje español.

El Usuario ingresa e informa al WazeLog donde se encuentra y a donde desea llegar.
Una conversación sería:

DrWazeLog: Bienvenido a WazeLog la mejor lógica de llegar a su destino.
Por Favor indíqueme donde se encuentra.
Usuario: Hola WazeLog estoy en Cartago.
DrWazeLog: Muy bien, ¿Cuál es su destino?
Usuario: San José.
DrWazeLog: Excelente, ¿Tiene algún destino intermedio?
Usuario: Tengo que pasar al Supermercado.
DrWazeLog: ¿Cuál Supermercado?
Usuario: Me gustaría AutoMercado
DrWazeLog: ¿Dónde se encuentra AutoMercado?
Usuario: Se ubica en Tres Ríos.
DrWazeLog: ¿Algún destino intermedio?
Usuario No
DrWazeLog: Su ruta sería Cartago, Taras, Tres Ríos, San Pedro, San José tiempo estimado de viaje 30 minutos.
Usuario: Muchas gracias por utilizar WazeLog.

La presente tarea se compone de cinco partes: la primera parte es la definición de una base de datos (hechos y reglas -el grafo-).

La segunda parte es la definición de las reglas para poder identificar las oraciones, sintagmas nominales y sintagmas verbales para lograr descomponerlos en palabras e identificar las palabras claves por ejemplo: Origen, destino, lugares, si, no, etc.

La tercera parte es la unión de las dos primeras partes en una interfaz de usuario donde el sistema reciba la oración del usuario y muestre la mejor ruta.

Es importante mencionar que el sistema puede no encontrar una palabra clave en la oración de entrada por lo que debe tener una oración conectora para poder proseguir con la conversación. Se sugiere utilizar listas pero no se limita a estas. **Se debe programar utilizando gramáticas libres de contexto.**

Cuarta parte interfaz gráfica en Java, debe realizarse una interfaz gráfica utilizando Java que realice una liga con el código de Prolog el cual será el motor de esta aplicación. En este caso la aplicación habilitara el grafo en pantalla y por medio de opciones del mouse se puede seleccionar el nodo inicial y el nodo final.

Quinta parte, aprendizaje al realizar la interfaz gráfica con Java este puede incluir nuevas rutas.

6. **Entregables**
   6.1. Código fuente comentado.
   6.2. Manual de usuario.
7. **Documentación**
   1. Se deberá entregar un documento que contenga:
      1.1. Manual de usuario: cómo ejecutar el programa.
      1.2. Descripción de los hechos y reglas implementadas.
      1.3. Descripción de las estructuras de datos desarrolladas.
      1.4. Descripción detallada de los algoritmos desarrollados.
      1.5. Problemas conocidos: En esta sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.
      1.6. Actividades realizadas por estudiante: Este es un resumen de las bitácoras de cada estudiante (estilo timesheet) en términos del tiempo invertido para una actividad específica que impactó directamente el desarrollo del trabajo, de manera breve (no más de una línea) se describe lo que se realizó, la cantidad de horas invertidas y la fecha en la que se realizó. Se deben sumar las horas invertidas por cada estudiante, sean conscientes a la hora de realizar esto el profesor determinará si los reportes están acordes al producto entregado.
      1.7. Problemas encontrados: descripción detallada, intentos de solución sin éxito, soluciones encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.
      1.8. Conclusiones y Recomendaciones del proyecto.
      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. 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 65% 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 tiene un valor de 15%, todos los miembros deben participar y todos deben conocer el código.
   4. Cada grupo recibirá una nota en cada uno de los siguientes apartados Código y Documentación.
   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 lógico y orientado a objetos**, 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 los puntos 1, 2 y 3 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 y la documentación 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 4 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 **Prolog/Java** utilizando el **paradigma de programación lógico/orientado a objetos**, en caso contrario 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 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**

Mendoza, A. (2019, 3 10). *Como conectar Java con Prolog*. Retrieved from Como conectar Java con Prolog: http://webcache.googleusercontent.com/search?q=cache:_BT89skHttUJ:ftp://190.114.210.136/IA/herramientas/swiprolog/conectar%2520Java%2520con%2520swi-Prolog/COMO%2520CONECTAR%2520JAVA%2520CON%2520PROLOG.pdf+&cd=11&hl=es-419&ct=clnk&gl=cr

Monferrer, T. E., Toledo, F., & Pacheco, J. (2001). *El lenguaje de programacion Prolog.* Valencia.

Swi. (2019, 3 10). *JPL: A bidirectional Prolog/Java interface*. Retrieved from JPL: A bidirectional Prolog/Java interface: http://www.swi-prolog.org/packages/jpl/