Instrucciones para la realización y entrega de prácticas

Realización de la práctica

- Las prácticas se deben realizar de forma individual.
- Se debe entregar un sólo fichero que *debe llamarse necesariamente* `code.pl`.
- Este fichero *debe empezar con una declaración de módulo*. Por ejemplo:

  ```
  :- module(_,_,[pure,assertions,regtypes]).
  ```

  para la parte de programación lógica pura (práctica 1), para que no se carguen predicados predefinidos, y:

  ```
  :- module(_,_,[classic,assertions,regtypes]).
  ```

  para la segunda parte del curso (práctica 2), para que se carguen además los predicados predefinidos ISO-Prolog y otros clásicos de Prolog.
- A continuación, *en el mismo fichero*, después de la declaración `:- module`, *hay que incluir un hecho*:

  ```
  author_data(Apellido1,Apellido2,Nombre,NumMatricula).
  ```

  Por ejemplo, si el nombre es *Ignacio Javier García Lombardía* con número de matrícula *D160125*, entonces sería necesario incluir el hecho:

  ```
  author_data('García', 'Lombardía', 'Ignacio Javier', 'D160125').
  ```

  Importante:
  - Como los nombres empiezan con mayúsculas, hay que usar comillas simples '…' como arriba para que sean leídos como átomos y no como variables.
  - Se pueden usar acentos y ñ, siempre que estén codificados en utf8.
- A continuación, *en el mismo fichero*, hay que implementar los predicados requeridos en el enunciado. El nombre de predicado, aridad y orden de los argumentos deben ser idénticos a los del enunciado. Si no el corrector automático no los encontrará y dará 0 puntos. Los predicados auxiliares (no pedidos en el enunciado) pueden tener cualquier nombre (distinto).
- Vuestro código debe poder ejecutarse y funcionar correctamente en Ciao Prolog, dado que es el sistema que ejecuta los tests en el sistema `Deliverit`.

Es sumamente importante seguir todos estos pasos al pie de la letra, ya que de no hacerse correctamente, el corrector puede rechazar la práctica.

Documentación

- El fichero `code.pl` debe estar bien documentado, lo que se debe hacer incluyendo en el mismo fichero `code.pl` *aserciones* y/o *comentarios legibles mecánicamente* del lenguaje Ciao Prolog. Esto permitirá generar automáticamente un manual/memoria procesando dicho fichero con el auto-documentador (`lpdoc`).
- Esta herramienta está incluida en la instalación de Ciao Prolog y se puede también correr desde Emacs, VSC, o el playground.
- En Moodle tenéis instrucciones específicas sobre cómo hacer todo esto y un ejemplo de código que tiene ya comentarios y tests (ver abajo), para practicar corriendo `lpdoc` sobre él y, en su caso, ejecutando los tests con Ciao.

En cuanto a la documentación en sí:

- El manual debe tener un título y empezar con una introducción, describiendo lo que hace el módulo.
- También deben estar documentados los predicados, describiendo los argumentos, y la semántica del predicado, es decir, cuándo se hace cierto.
- Es interesante también incluir información sobre la aproximación y decisiones tomadas, así como cualquier comentario adicional que se estime oportuno.
- Asimismo, se pueden describir consultas realizadas al programa y las respuestas obtenidas.

Vuestras Pruebas / Tests

- Vuestras consultas de prueba y respuestas esperadas se pueden codificar como un conjunto de aserciones `:- test` en el código. Ello es muy recomendable y os puede ahorrar trabajo. Se valorará (sólo para subir nota) el uso de dichas aserciones `:- test` para definir casos de prueba y comprobar el funcionamiento de los predicados.
- En todo caso estos tests se deben incluir en el mismo fichero `code.pl`.
- Como comentamos, hemos dejado en Moodle instrucciones y ejemplos.

Nota: Los tests que `Deliverit` aplica a vuestro código son independientes de los que pueda haber en vuestro código. Si tenéis tests en vuestro código, como os recomendamos, éstos son para uso vuestro y documentación, y podéis ejecutarlos en Ciao Prolog, pero `Deliverit` no los ejecutará.

Otras recomendaciones

- Os recomendamos fuertemente primero desarrollar y probar directamente en Ciao Prolog vuestros predicados, y comprobar que os funcionan para diferentes tipos de consulta o modos, con argumentos más o menos instanciados, etc.
- Recordad *pedir soluciones adicionales* (usando `;` en el top level), ya que los tests de *Deliverit* suelen comprobar si hay una segunda solución o más.
- Es importante comprobar localmente que al aplicar el auto-documentador a vuestro fichero `code.pl` se genera correctamente el manual/memoria.
- Para todas estas pruebas, recomendamos instalar Ciao Prolog localmente y usar preferiblemente el entorno integrado con Emacs o VSC, usando el depurador, el documentador, etc. Ver las instrucciones en Moodle, transparencias de clase y manuales. También podéis usar el playground de Ciao, que no necesita instalación pero es algo más limitado.

Comprobación y entrega en Deliverit

- Una vez estéis satisfechos con cómo os funciona vuestro código localmente, podéis subirlo a Deliverit para ver si pasa las comprobaciones y tests allí.
- En Moodle, junto al enunciado de la práctica, tenéis el enlace a *Deliverit*. Si no habéis entrado nunca antes en `Deliverit`, introducid el número de matrícula, sin clave, seleccionad "olvidado contraseña," y recibiréis una contraseña por correo.
- No es necesario que vuestro código pase todos los tests, ni que estén todos los predicados, para que la práctica cuente como entregada y se califique.
- Podéis subir el código a *Deliverit* cuantas veces queráis hasta la fecha límite de entrega. La **última subida** se considerará la **entrega oficial**.

Calificación

- El sistema *Deliverit* os da un número entre 0 y 1, que se corresponde con la fracción de casos de prueba que pasa vuestro código.
- Puntuación: en general el código es el 85 % (8.5 puntos) de la nota. El 15 % restante (1.5 puntos) es la memoria/manual auto-generada.

Nota: El número de casos pasados con éxito es una parte importante de la calificación, pero se mira además el estilo del código, los comentarios / manual / memoria, etc., y se hacen otros tests. En todo caso el valor que os da *Deliverit* es una aproximación razonable de la corrección de vuestro código.

Es posible que –siempre para ayudaros y avisando– cambiemos algún test durante el periodo de entrega si encontramos algún error o inconsistencia en ellos.