Primera práctica (Programación lógica pura)

Fecha de entrega: viernes 20 de marzo de 2020

EDIFICIOS DE VIVIENDAS

Enunciado

En esta práctica representaremos edificios de viviendas, con varios niveles, y con varias viviendas en cada nivel. La estructura de datos que utilizaremos será una lista formada por los diferentes niveles, donde cada nivel es a su vez una lista de las viviendas de ese nivel. Los valores almacenados representan el número de personas que viven en cada vivienda, para lo que utilizaremos números naturales, en notación de Peano. Con esta representación, el término:

```
[ [ s(0), s(s(s(0))) ], [ 0, s(s(0)) ] ]
```

representa el edificio:

	Vivienda 1	Vivienda 2
Nivel 2	`0`	`s(s(0))`
Nivel 1	`s(0)`	`s(s(s(0)))`

Los edificios pueden tener un número arbitrario de niveles y de viviendas en cada nivel. Las listas en las respuestas deben mantener el orden en el que aparecen en la estructura que describe el edificio. Las tareas a realizar son las siguientes:

1. Definir el tipo:

   `basic_building(X): X` *es un edificio, que contiene números naturales. Debe tener al menos un nivel y cada nivel al menos una vivienda.*

2. Definir el tipo:

   `building(X): X` *es un edificio como el anterior pero además todos los niveles deben tener el mismo número de viviendas.*

3. Definir el predicado `level/3` tal que:

   `level(X,N,C): C` *es el nivel* `N`*-ésimo del edificio* `X` *(la lista con todas las viviendas de ese nivel).*

4. Definir el predicado `column/3` tal que:

   `column(X,N,C): C` *es la lista formada por las viviendas* `N`*-ésimas de todos los niveles del edificio* `X`*. Es decir, si* `X` *está instanciada al edificio ejemplo de arriba, la consulta* `column(X,s(0),C)` *respondería* `C = [s(0), 0]`.

5. Definir el predicado `columns/2` tal que:

   `columns(X,C): C` *es la lista de las columnas de viviendas del edificio.*

6. Definir el predicado `total_people/2` tal que:

   `total_people(X,T): T` *es el número total de personas que viven en el edificio* `X`.

7. Definir el predicado `average_by_apartment/1` tal que:

   `average(X,A): A` *es la media de personas que viven en cada vivienda del edificio, redondeada al número natural más cercano.*

   PUNTOS ADICIONALES (sólo subir nota):

8. Ejercicio complementario en 'programación alfabetizada' ('literate programming'):

   Se darán puntos adicionales a las prácticas que realicen la documentación de dichos predicados insertando en el código aserciones y comentarios del lenguaje Ciao, y entregando como memoria o parte de ella el manual generado automáticamente a partir de dicho código, usando la herramienta `lpdoc` del sistema Ciao. Se recomienda intentar escribir este manual de forma que sustituya completamente a la memoria.

9. Ejercicio complementario en generación de casos de prueba:

   También se valorará (sólo para subir nota) el uso de aserciones `test` que enumeren casos de prueba para comprobar el funcionamiento de los predicados.

Instrucciones adicionales

- Las prácticas se realizarán en grupos de 3 alumnos.
  - Todos los miembros de un grupo deben pertenecer a un sólo grupo de teoría (que ha se ser en el que estén matriculados).
  - Una vez formados estos grupos para la primera práctica, quedarán consolidados para todas las prácticas posteriores.
  - Se elegirá a un portavoz de grupo que será responsable de subir a Moodle las entregas. El resto de alumnos del grupo no deben subir nada a Moodle. La portavocía del grupo, una vez consolidada tras la entrega de la primera práctica, se mantendrá para futuras prácticas.
- La práctica debe poder ejecutarse y funcionar correctamente en el entorno Ciao Prolog.
- Se recuerda que debido a que esta práctica repasa aspectos de la programación lógica pura no está permitido el uso de recursos de ISO-Prolog que van más allá, tales como, p.ej., predicados metalógicos, negación por fallo, aritmética con `is/2`, cortes, etc.
- Debe utilizarse exactamente el nombre de predicado, aridad y orden de los argumentos que vienen dados en el enunciado, ya que son los que se utilizarán en las pruebas de corrección. Los predicados auxiliares (no pedidos en el enunciado) pueden tener cualquier nombre, aridad y número de argumentos.

- Todos los predicados deben estar correctamente documentados en el código y en la memoria, indicando el nombre del predicado, los argumentos, y una descripción en lenguaje natural y/o una fórmula lógica indicando la semántica del predicado (es decir, cuándo se hace cierto).

Entrega de las prácticas

Las prácticas se entregarán a través del Aula Virtual.

El portavoz del grupo será el encargado de realizar la entrega a través de Moodle.

La entrega debe constar de los siguientes ficheros:

1. Fichero con el código fuente

   El archivo debe llamarse `codigo.pl`

   Dentro de este archivo se indicará quienes son los componentes del grupo de prácticas mediante la inclusión de tantos hechos de tipo `alumno_prode/4` como alumnos haya en el grupo.

   Estos hechos deben tener la forma:

   `alumno_prode(`*Apellido2*`,`*Apellido1*`,`*Nombre*`,`*NumMatricula*`).`

   Indican al corrector automático los datos de cada componente del grupo. Por ejemplo, si uno de los componentes del grupo se llamase *Ignacio Javier García Lombardía* con número de matrícula *D160125*, entonces sería necesario incluir el hecho

   ```
   alumno_prode('Garcia', 'Lombardia', 'Ignacio Javier', 'D160125').
   ```

   IMPORTANTE:

   - Como los nombres empiezan con Mayúsculas, es importante incluir las comillas para marcar que los nombres sean átomos y no variables.
   - No incluir caracteres acentuados ni la ñ ni ningún otro carácter "extraño" en el nombre y/o los apellidos.

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

2. Fichero con la memoria de la práctica.

   El archivo debe llamarse `memoria.pdf` o `memoria.txt`

   La memoria debe indicar también claramente los nombres de los alumnos que forman el grupo, así como sus números de matrícula. La memoria debe incluir:

   a) el código, junto con la explicación y justificación de las decisiones;
   b) las consultas realizadas con el programa y las respuestas obtenidas con dichas consultas; y
   c) opcionalmente, cualquier comentario aclaratorio que se estime oportuno.

   Como se comentó anteriormente, la memoria también se puede generar a partir del código con la herramienta lpdoc y las consultas del apartado b) se pueden expresar como un conjunto de aserciones `test`.