Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza
1542

Práctica 7 (opcional)

Tecnología de Programación

Grado en Ingeniería Informática

Objetivos

- Aplicar la potencia de un lenguaje funcional a la resolución de problemas complejos.
- Utilizar estructuras avanzadas de los lenguajes funcionales como las listas por comprensión.
- Intuir cómo se trasladan los conceptos de complejidad algorítmica a la programación funcional.

Nota: La resolución de esta práctica es opcional. Esto significa que se puede aspirar a la máxima nota en prácticas sin la necesidad de realizar esta. Sin embargo, la realización de esta práctica podrá compensar una nota baja en otra práctica anterior, sin superar en ningún caso la nota máxima de prácticas.

1. Juego de lógica

Existen numerosos **juegos de lógica** que se pueden resolver mediante la programación funcional en Haskel (como por ejemplo el de las 8 reinas). Una de las ventajas de un lenguaje funcional es que permite explorar con relativa facilidad para el programador todo el espacio de soluciones a dichos juegos de lógica.

Tarea: Se pide elegir un juego de lógica (excepto el de las n reinas y el de las garrafas de agua). Busca en internet la descripción del juego de lógica si es necesario. Asegúrate que tu juego de lógica cumple con los siguientes requisitos:

- Es un juego para **un solo jugador**.
- Es un juego que es **matemáticamente simple** (no requiere conocimientos avanzados de estadística o probabilidad).
- Es un juego que **no depende de elementos aleatorios** como dados, cartas o monedas que se lanzan al aire.
- El juego **no depende del lenguaje** (no es un juego de palabras que requiera un diccionario) ni de la semántica (no requiere conocer el significado de un texto o figura).
- El objetivo o meta del juego **no es ambiguo**, y es fácil de identificar, medir y analizar.

En un documento (máximo una página, en formato `.pdf`) describe del modo más formal posible en qué consiste dicho juego de lógica, y cuál es su objetivo. Evita ambigüedades en la descripción. Utiliza diagramas o figuras si es necesario.

2. Exploración funcional del espacio de soluciones

Las potentes estructuras de los lenguajes funcionales permiten explorar **todo el espacio de soluciones** de un juego de lógica con pocas líneas de código. Dicha exploración puede dar como resultado todas las posibles soluciones al juego de lógica que has elegido y descrito.

Tarea: Implementa una función en Haskell que muestre todas las posibles soluciones al juego de lógica. **Parametrízalo** todo lo que sea posible. Por ejemplo, habría que resolver el problema de las n reinas para cualquier n (no fijado para 8). Otro ejemplo: en el juego de las torres de Hanoi, habría que resolverlo para n torres y m piezas (en lugar de 3 torres y 3 piezas). Implementa todas las funciones auxiliares que consideres necesarias. Para cada función que implementes, pon en un comentario qué es lo que hace.

Para juegos que dependan de un número de pasos a seguir (como el de las torres de Hanoi o las garrafas de agua) el potencial número de soluciones es infinito (siempre puedes volver a un estado anterior). Para dichos problemas, en lugar de devolver todas las soluciones (infinitas) deberás devolver **todas las soluciones para un número determinado de pasos** (finitas) que le pasarás como entrada a la función.

Haz una función `main` que pruebe tu implementación bajo ciertos casos y ciertos parámetros de entrada. Incluye en comentarios qué es lo que estás probando y por qué.

3. Código descargado de internet

Está permitido buscar soluciones Haskell a un juego de lógica en **internet** (las hay, y muchas). En ese caso, el alumno deberá centrar su trabajo en conseguir que el código Haskell descargado compile y que todas las funciones y tipos de dato del código tengan comentarios escritos por el alumno que expliquen exactamente qué es lo que hace dicha función (el esfuerzo del alumno, en ese caso, consiste en entender el código descargado). Es obligatorio que, si el código está descargado de internet, **esté indicado tanto en el documento pdf como en todos los archivos Haskell entregados**, incluyendo la URL completa de el origen del código Haskell.

Para una práctica basada en código descargado de internet, la nota máxima será de un 7 sobre 10. Si en un trabajo basado en código descargado de internet, el trabajo no se indica convenientemente que lo es ni se incluye la URL origen de dicho código, se considerará que es un plagio y la nota será de 0 sobre 10. Evidentemente, el código descargado de internet debe cumplir con los requisitos de la práctica, adaptándolo si es necesario.

Entrega

Como resultado de esta práctica deberás entregar los siguientes archivos:

- `descripcion.pdf` donde incluyas la descripción formal del juego de lógica.
- Todos los archivos `.hs` que hayas necesitado (si has necesitado más de uno).
- Un archivo `Main.hs` donde se encuentre la función `main` que haga la prueba expresada anteriormente.

Todos los archivos de código fuente solicitados en este guión deberán ser comprimidos en un único archivo `zip` con el siguiente nombre:

- `practica7_<nip1>_<nip2>.zip` (donde `<nip1>` y `<nip2>` son los NIPs de 6 dígitos los estudiantes involucrados) si el trabajo ha sido realizado por parejas. En este caso sólo uno de los dos estudiantes deberá hacer la entrega.
- `practica7_<nip>.zip` (donde `<nip>` es el NIP de 6 dígitos del estudiante involucrado) si el trabajo ha sido realizado de forma individual.

El archivo comprimido a entregar no debe contener ningún fichero aparte de los fuentes que te pedimos: ningún fichero ejecutable o de objeto, ni ningún otro fichero adicional.

La entrega se hará en la tarea correspondiente a través de la plataforma Moodle: `http::/moodle.unizar.es`.