Práctica 4

Enunciados

Cada alumno tendrá que realizar un ejercicio, al menos el contenido mínimo que se plantea. Se puede hacer en grupos de 2 o 3 si se opta por contenidos adicionales suficientes. También se puede proponer otra práctica diferente, pero debe ser aceptada por el profesor.

La asignación de los ejercicios será ante las peticiones del alumno(s), tras mandar un mensaje al profesor solicitándolo, indicando el grupo que se propone y la tarea a realizar. Para evitar un intercambio de mensajes excesivo, se recomienda solicitar varios ejercicios en orden. La asignación se actualizará en el Moodle cada 2 o 3 días basándose en 1) las notas de las prácticas anteriores y 2) orden de llegada.

- El pintor informático.
- Para no perderse.
- El poeta informático.
- El maldito corrector.
- A la deriva(da).
- Jugando con las palabras.
- Generador de Sudokus.
- Hora punta.
- El colorido fontanero.

Como apéndices se presentan informaciones adicionales de cada uno de ellos. Con esta información se debe trabajar, correspondiendo a los alumnos tomar las decisiones que no estén claramente indicadas en el enunciado. En todo caso, el programa tendrá un `main` ejecutable con unos datos o parámetros de prueba o comportamiento interactivo. Se pueden utilizar bibliotecas de Haskell y datos o ficheros obtenidos de internet. Por ejemplo, aquí hay un fichero de texto con un diccionario de castellano (pueden usarse versiones simplificadas).

Entrega

La entrega será mediante correo electrónico al profesor de la asignatura. Debe contener:

- El código `PF_Pratica4.hs` del programa principal (con un `main`) y los ficheros de los posibles módulos adicionales. Se debe ejecutar por sí mismo, sin llamar a ninguna función específica. También tendrá incluidas las opciones de compilación.
- Los ficheros adicionales con datos utilizados en el programa y/o pruebas, salvo si son los suministrados en el Moodle.
- Un fichero `NombreApellidos.pdf` con el nombre del alumno y un manual sucinto del código desarrollado, donde aparezca información clara de todas las decisiones tomadas: una explicación detallada de lo que hace el programa, cómo se usa, qué limitaciones tiene, las pruebas realizadas y cómo se cambian si fuera necesarios. No exime de que el código esté adecuadamente comentado.

Las calificaciones se basarán en diversos criterios: calidad del código, uso adecuado de los elementos principales del curso (tipos algebraicos, orden superior, listas por comprensión, mónadas, etc.), documentación, cuanto se ha hecho del contenido adicional, etc.

Como de casi todas las prácticas que os proponemos que simulan un problema o juego habitual, existen muchas versiones y páginas en internet. También podemos haberlas propuesto en años

anteriores. Utilizarlas para inspirarse es natural y aconsejable, pero la copia con cambios cosméticos, sin saber exactamente lo que se hace, supone una violación del código ético del estudiante y de la Ley 3/2022, de 24 de febrero, de convivencia universitaria, así como la normativa propia de la UPM, que califica el fraude como falta grave (artículo 12). Se implantarán mecanismos de detección de copias.

Todas las prácticas recibidas antes del viernes 26 de mayo serán evaluadas con un breve informe que indica las posibles mejoras (bien porque no son aptas bien porque la nota obtenida es baja) y podrán reenviarse. En todo caso, las prácticas deben entregarse hasta el día anterior a la fecha del examen de la asignatura para ser evaluadas en la correspondiente convocatoria y su evaluación positiva será condición imprescindible para superar la asignatura.

Jugando con las palabras

La mayor parte de los pasatiempos típicos de los periódicos o los juegos de mesa de palabras se basan en manejar palabras. Al resolverlos, se utilizan todo tipo de conocimientos sobre el castellano y no sólo sobre las palabras y sus significados. Por ejemplo, no es posible aceptar cualquier palabra, ya que hay combinaciones de letras que sabemos que no son posibles en castellano: las consonantes que aparecen como final de sílaba son únicamente L, N, S, R; sólo aparecen dos consonantes empezando una sílaba cuando la segunda es una L o R (cuidado, la CH en castellano es una única letra; por tanto solamente pueden aparecer tres consonantes seguidas con un final de consonante y un grupo doble de principio de sílaba (con la excepción de que la N se torna en M seguida de B, P), etc.

El programa jugará al conocido juego del Scrable, o, al menos, simulará una jugada. Tendrá que establecer un conjunto de reglas para generar/decidir cuando una palabra es aceptable en castellano (aunque no signifique nada). Por ejemplo: *fatón*, *mute* o *carba* suenan aceptables, mientras *fatzon*, *mte* o *carbma* no lo son. También tendrá que realizar un generador de palabras: dada una serie de letras (por ejemplo, en una *string*,) que son las palabras de un jugador y otra serie (también en una *string*, que puede ser vacía) con letras usadas en un tablero, buscará la palabra de máxima longitud que pueda formarse con las letras del jugador y como mucho una de las letras del tablero, de manera que parezca una palabra del castellano. Ejemplos de uso:

Mano	Tablero	Propuestas posibles
lvroeu	e b	revuelo beluero
ldrnoc	a w	ladron lacron
bedwo	a e	bado cebo

Contenido mínimo

Simular una jugada del Scrable: Obtener palabras por métodos sencillos (ejemplo, combinaciones), filtrarlas con algunas reglas sencillas y proponerlas al usuario.

Contenido ampliado

Hacer un programa interactivo que muestre la situación actual (mano y palabras en el tablero) y proponga al usuario palabras para que elija. Opcionalmente se puede programar una versión simple del Scrable.

También se pueden ampliar las reglas y comprobarlas en un diccionario almacenado en un fichero (ejemplo).

El poeta informático

El poeta informático quiere realizar un programa que le auxilie en su tarea de estudio de los grandes poetas clásicos. En particular, el poeta quiere poder determinar automáticamente el tipo de poema que conforman unos versos dados. El tipo de poema queda determinado por su métrica (número de sílabas de cada verso), su rima (asonante o consonante) y su número, que, combinado con la métrica, dan una estructura del poema (sonetos, cuartetos, pareado, etc.).

Por ejemplo, el siguiente poema escrito por Lope de Vega:

> Un soneto me manda_hacer Violante,
> que_en mi vida me_he visto_en tanto_aprieto;
> catorce versos dicen que es soneto,
> burla burlando van los tres delante.
>
> Yo pensé que no hallara consonante
> y estoy a la mitad de otro cuarteto,
> mas si me veo en el primer terceto,
> no hay cosa en los cuartetos que me espante.
>
> Por el primer terceto voy entrando
> y parece que entré con pie derecho
> pues fin con este verso le voy dando.
>
> Ya estoy en el segundo y aun sospecho
> que voy a los trece versos acabando;
> contad si son catorce y está hecho.

está escrito en endecasílabos (11 sílabas por verso - ¡cuidado¡: en el ejemplo aparecen sinalefas, es decir, uniones entre una palabra que acaba en vocal y otra que empieza por ella a efecto de las sílabas; se han marcado las primeras con un subrayado -), su rima es consonante y la estructura es de soneto (14 versos endecasílabos, organizados en dos cuartetas y tres tercetos con rima en pares y rima consonante). Mas información, por ejemplo, en https://www.poemas-del-alma.com/blog/especiales/nombres-estructuras-poeticas

El programa pedido debe identificar estas características a partir de un poema, declarando los tipos adecuados.

Contenido mínimo

Usar una variedad de estructura de poemas limitada (pero no excesivamente trivial), incluir palabras con todos los acentos explícitos (es decir, cuando una vocal va acentuada aparecerá expresamente con tilde), admitir un uso limitado de la *y,* asumir rimas entre todos los versos y olvidarse de cuestiones como la sinalefa anteriormente comentada.

Contenido adicional

Limitar las restricciones anteriores: Por ejemplo, aumentar la variedad de poemas (o hacer un tipo para definirlas), detectar las sílabas sin tildes explícitas (es decir, detectar diptongos, iatos, triptongos, … con las reglas del castellano), permitir rimas más complejas como la del soneto, etc.

También se pueden proponer extensiones alternativas, como inventar poemas de unas ciertas características a partir de un diccionario.

Para no perderse

Se quiere optimizar una nueva red de carreteras en un área con una serie de ciudades C = {c_1, c_2, …, c_n} conectadas entre sí (algunas de ellas, pero todas están conectadas) por vías V que tienen un peso, por ejemplo, la distancia entre ellas: V={…(c_i, c_j, p_k)…} . La solución será elegir entre todas las carreteras un subconjunto T ⊂ V, que permita conectar todas las ciudades y que tenga coste mínimo. De esta forma, se podrá mejorar la red de forma eficiente y óptima en coste.

Sugerencias:

Representar las ciudades C y las carreteras V con un grafo ponderado y buscar soluciones al problema.

Si se requiere alguna pista adicional, contactar con el profesor.

Contenido mínimo

Definir un TAD para grafos ponderados. A partir de sus operaciones, desarrollar el programa como se ha indicado, probándolo con juegos de datos significativos. El algoritmo no debe tener una complejidad exponencial. Si se utilizan algoritmos conocidos, identificarlos por su nombre en el código y la documentación.

Contenido adicional

A partir de vuestra solución, generar laberintos sin ciclos y con solución única entre la entrada y la salida suficientemente largos.

Sudokus

Es un puzle muy famoso que no necesita descripción. Se recomienda mirar en http://www.websudoku.com para ver ejemplos.

Contenido adicional

Solo se acepta para dos personas. Se desarrollará un programa que *genere* Sudokus, clasifique su dificultad y, adicionalmente, permita jugar con ellos. La funcionalidad sería parecida a la de la aplicación de la página web mencionada.

El maldito corrector

Se trata de implementar una aplicación tipo Teshaurus que se utiliza en los correctores de aplicaciones muy comunes (procesadores de texto, mensajería, etc.). En primer lugar, se pide desarrollar en Haskell un tipo para representar los trie (árboles cuya información se construye con la cadena que va de la raíz a las hojas). Este tipo permitirá construir un diccionario de palabras simples del castellano (implementado con un trie, donde en las hojas aparece al menos la característica de la palabra [sustantivo/verbo, género, etc.]) y ante una palabra nueva debe determinar si posiblemente pertenece al castellano o no. En el diccionario no aparecen las palabras derivadas, esto es ni los plurales, ni los cambios de género, ni los participios, ni los tiempos verbales, etc.

Contenido mínimo

Realizar el programa indicado con reglas del castellano simples.

Contenido adicional

Mejorar las reglas indicadas (incluir gerundios, formas verbales, incluso algún verbo irregular). Ampliarlo a la corrección de textos completos, señalando las nuevas palabras con alguna marca (aportando opciones) y aquellas que no se reconocen.

El pintor informático

Algunos pintores utilizan elementos geométricos como base de sus pinturas. Por ejemplo, el pintor ruso-polaco Kasimir Malévich tiene series de cuadros con líneas, rectángulos y cuadrados rotados y colores básicos, en un estilo llamado suprematismo.

Situaciones similares se dan en los cuadros de Mondrian (rectángulos y cuadrados de colores básicos), Kandinsky (círculos, triángulos, líneas, …), Vasarely (espirales, cubos, …), Miró (dibujos a mano de espirales, estrellas, polígonos, medias lunas, etc.), cada uno con sus propias gamas de colores.

Contenido mínimo

El problema consiste en realizar un programa que dados unos cuantos parámetros básicos (número de figuras, tamaños límite, dimensiones, grados de los giros, etc.) "pinte" nuevos cuadros de Malévich con reglas que se “asemejen” a los originales de pintor.

Contenido adicional

Ampliar los pintores (por ejemplo, con las propuestas indicadas) y determinar sus características habituales (número de elementos de cada tipo, colores habituales, disposición habitual, etc.)
Bautizar los cuadros al estilo del pintor y generar galerías "ficticias" en html.

A la deriva(da)

En la práctica 2 os propusimos un tipo para representar expresiones. Se trata ahora de modificarlo, eliminando el condicional (`if`) y extendiendo las fórmulas matemáticas a tratar (divisiones, logaritmos, exponenciales, raíces, trigonometría, …) para poder definir funciones en una o varias variables.

A partir de esas funciones se trata de ofrecer un kit de operaciones para trabajar con ellas: aplicarlas a unos valores, derivarlas, aplicar la derivada a unos valores, …

Contenido mínimo

Definir funciones en una variable y derivarlas. Las derivadas resultantes deben de simplificarse al menos con reglas básicas (por ejemplo, que no haya dos monomios del mismo exponente, que no se sumen ceros, etc.)

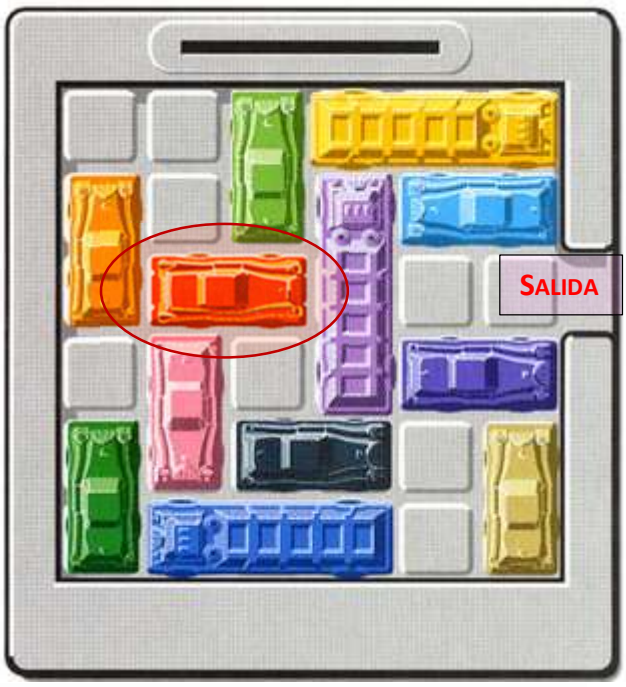
Las funciones deben de mostrarse de la forma más legible posible (clase `Show`)

Contenido adicional

Ofrecer un kit interactivo que lea funciones de la pantalla, permite derivarlas, aplicarlas, operar con ellas, … También se pueden mejorar las simplificaciones.

Hora punta

Rush Hour es un conocido rompecabezas de bloques deslizantes inventado por Nob Yoshigahara en la década de 1970. Simula un parking donde se aparcan coches (que ocupan dos posiciones) y camiones (que ocupan tres). El objetivo es conseguir que uno de los coches (en la imagen el rojo, siempre en la tercera fila) llegue a la salida, moviendo los otros vehículos. Un movimiento supone desplazar un vehículo una posición a la izquierda, a la derecha, arriba o abajo, si hay espacio para ello.



Existen versiones físicas del juego y también digitales. Está disponible de forma gratuita en prácticamente todos los dispositivos. También puede jugarse online en:

https://www.thinkfun.com/rush-hour-online-play/

Contenido adicional

Solo se acepta para dos personas. Se desarrollará un programa que dado una situación inicial (comienzo del juego) lo clasifique por su dificultad: principiante, intermedio, avanzado y experto. Para ello se propondrá una fórmula (polinomio) en función de características de la solución mínima, por ejemplo, número de pasos, proporción de piezas movidas, grado de simetría, número de piezas en la situación inicial, etc.

En el Moodle podéis encontrar una base de datos (fichero RushHour.txt) con muchos ejemplos para probarla. Se muestra a continuación cómo está construida (sugerencia, definir el `read` de la clase `Read` del tablero). Ya están clasificados, de manera que pueden servir para probar la solución propuesta y variarla en función de los resultados.

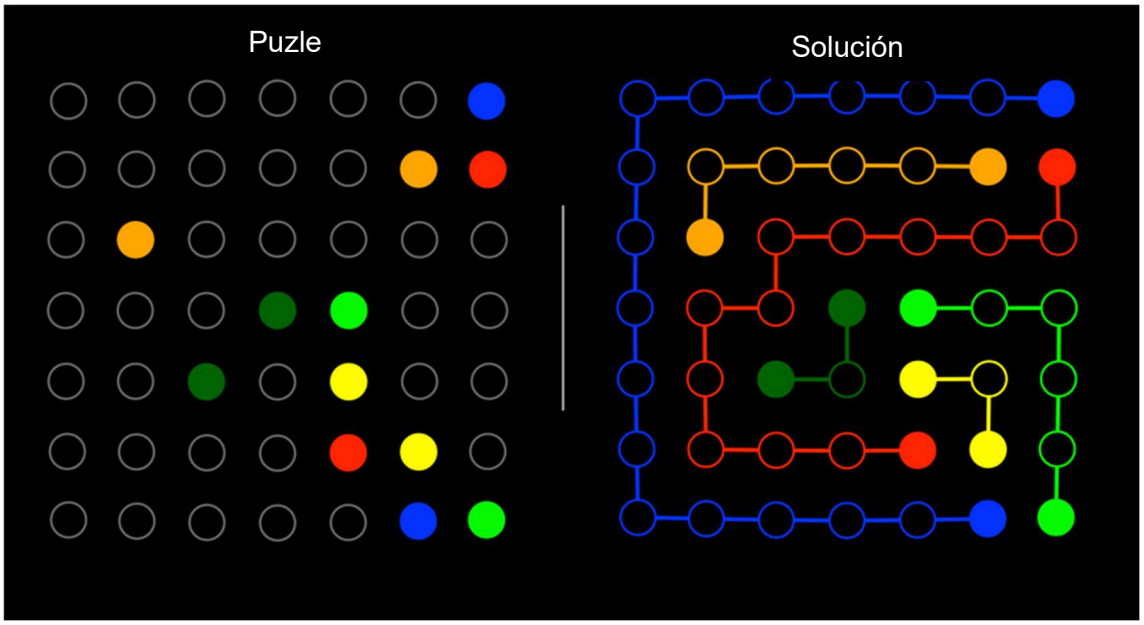
Si varios estudiantes o grupos optan por esta práctica, se sugiere que compartan los nuevos ejemplos vía el Foro de Moodle de la asignatura.

Leyenda del fichero	
o	Posición vacía
A	Coche que mover (rojo en la imagen)
B - Z	Resto de vehículos
Cada línea tiene una cadena con 36 caracteres, con representación del tablero sin resolver. Cada 6 elementos representan una fila.	

El colorido fontanero (Flow)

Se trata de resolver un conocido juego donde la posición inicial en un tablero de N×M contiene varios puntos de colores, un par por cada color. Se trata de conectar colores iguales para formar una tubería. Se deben completar todos los colores y cubrir el tablero entero de tuberías, que no pueden cruzarse ni superponerse. El juego está disponible en casi todos los dispositivos de forma gratuita y se puede jugar online en https://es.y8.com/games/flow_mania

Las soluciones suelen ser visualmente muy bonitas. Es conocido que el problema es NP-Completo, de manera que las soluciones (únicas) al puzle son complejas y de alto coste computacional.



En Moodle se ofrecen algunos ejemplos de prueba (fichero Flow.txt), pero se aconseja que se añadan más. Si varios estudiantes o grupos optan por esta práctica, se sugiere que compartan los nuevos ejemplos vía el Foro de Moodle de la asignatura.

Contenido mínimo

Modelar los tableros y escribir un programa que dado un tablero inicial encuentre la solución. Añadir pruebas a las ya propuestas.

Contenido adicional

Añadir visualizaciones a la solución, generador de puzles, permitir jugar, etc. Hay múltiples variaciones que pueden también implementarse (muros, tableros irregulares, etc.)