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.
- Buscando su camino (en grafos dirigidos).
- El poeta informático.
- El maldito corrector.
- Comprensión de ficheros.
- Jugando con las palabras.
- Jugando con los números.
- Sudokus.

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.
- Un fichero `NombreApellidos.txt` con el nombre del alumno e 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.

Todas las prácticas recibidas antes del viernes 27 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 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 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 anagramas: un anagrama es una palabra formada con todas las letras de otra, pero en otro orden de manera que parezca una palabra del castellano. Además, debe ser suficientemente distinta de la otra (no es suficiente cambiar dos vocales). Así, las palabras *veleuro*, *roldan* y *bidorra* son anagramas de *revuelo*, *ladrón* y *abridor*. Finalmente realizar un resolvedor de anagramas, que construya palabras aceptables y las compruebe en el diccionario.

Contenido mínimo

Obtener palabras por métodos sencillos (ejemplo, permutaciones), filtrarlas con algunas reglas sencillas y comprobarlas con el usuario por pantalla.

Contenido ampliado

Construir palabras a partir de las reglas indicadas, ampliar dichas reglas y comprobarlas con un diccionario en un fichero.

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 hechos.

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.

Buscando su camino (en grafos dirigidos)

Una compañía telefónica necesita, para optimizar su tendido de cables de fibra óptica, un algoritmo sobre grafos acíiclicos dirigidos que permita calcular todos los caminos de una longitud fijada entre un nodo origen y uno destino.

El programa deberá proponer un tipo para estos grafos, sus operaciones más habituales y necesarias y el algoritmo pedido.

Contenido mínimo

Desarrollar el programa como se ha indicado, probándolo con juegos de datos significativos.

Contenido adicional

Añadir nuevas prestaciones: Pesos en los arcos del grafo, seleccionar el óptimo, evaluar las precondiciones (esto es, que es acíclico), menú interactivo, etc.

Jugando con los números

Es un juego popular que se ha utilizado en televisión en numerosos países. Dada una lista de cifras positivas (por ejemplo [1, 3, 7, 10, 25, 50]) se trata de encontrar la expresión infija con valor más próximo a un valor positivo prefijado (por ejemplo, 765) usando las operaciones +, -, *, ÷ aritméticas en los enteros.

Contenido mínimo

Realizar un programa que determine una expresión que calcule el número pedido (en el ejemplo, (25-10) ∗ (50+1), aunque hay otras 779 soluciones, aunque si el número a calcular es 831 no hay soluciones.

Contenido adicional

El espacio de búsqueda crece mucho si la cantidad de números inicial es muy alta. Si el programa combina la generación con la evaluación se reduce el espacio de búsqueda. Las más simples son evitar expresiones conmutativas (3*25 = 25*3) o triviales (50*1 = 50). También se pueden usar otras propiedades aritméticas, “memorizar” cómputos ya hechos o partir del valor inicial. Calcular los tiempos de ejecución de las versiones y visualizarlas y compararlas con la versión más sencilla.

Sudokus

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

Contenido mínimo

Desarrollar un programa Haskell que modele los Sudokus y los resuelva. Se valorará que se tomen decisiones para acotar el espacio de búsqueda y no se resuelva mediante mera fuerza bruta probando todas las opciones.

Contenido adicional

Desarrollar 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, marcando 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 holandés Piet Mondrian tiene series de cuadros con composiciones de cuadros, líneas y colores básicos.

Situaciones similares se dan en los cuadros de Malevich (líneas, rectángulos y cuadrados rotados), 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 rectángulos y/o figuras, tamaños límite, dimensiones, etc.) "pinte" nuevos cuadros de Mondrian con reglas que se “asemejen” a los originales de Mondrian.

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.

Compresión de ficheros

El algoritmo de Hoffman permite comprimir ficheros (por ejemplo, de imágenes). Consiste en asignar códigos de longitud variable a los caracteres en función de su frecuencia de aparición.

Considérese el texto:

ABRACADABRAPATADECABRA

Se calculan las frecuencias de cada elemento y se ordenan de menor a mayor: [(E,1),(T,1),(P,1),(D,2),(C,2),(R,3),(B,3),(A,9)]

Se construye el árbol de Huffman (de la siguiente forma:

- Se convierte cada elemento de la lista anterior en un árbol (será una hoja)
- Hasta que la lista tenga un único elemento:
    - Sacar los dos primeros árboles de la lista, formar una nueva rama con ellos y colocar dicha rama en el lugar adecuado de la lista. El lugar adecuado consiste en mantener ordenada la lista respecto a la suma de frecuencias de cada árbol.
    - Los códigos se asignan en función del camino que haya que recorrer para encontrar el elemento correspondiente en el árbol. En dicho recorrido, si se toma una rama a la izquierda, se considera un 1 y si es a la derecha, un 0.

De esa forma, los códigos asignados serían:

A=1, B=001, R=010, C=0000, D=0001, P=0111, T=01100, E=01101

Contenido mínimo

Desarrollar funciones que comprima/descompriman un fichero en otro con el algoritmo indicado.

Contenido adicional

Construir un programa interactivo que muestre un menú y admita las siguientes posibilidades:

- Comprimir/Descomprimir ficheros
- Encriptar/Desencriptar ficheros según una determinada clave
- Visualizar contenidos de ficheros (deteniendo la visualización en cada pantalla y permitiendo el avance y retroceso entre pantallas