Algoritmos y Estructuras de Datos
Guía de ejercicios de preparación para el 2do examen parcial - parte 1 (versión 2)

Temas: Archivos. Estructura cola, pila, lista, lista doblemente enlazada. Combinación de estructuras.

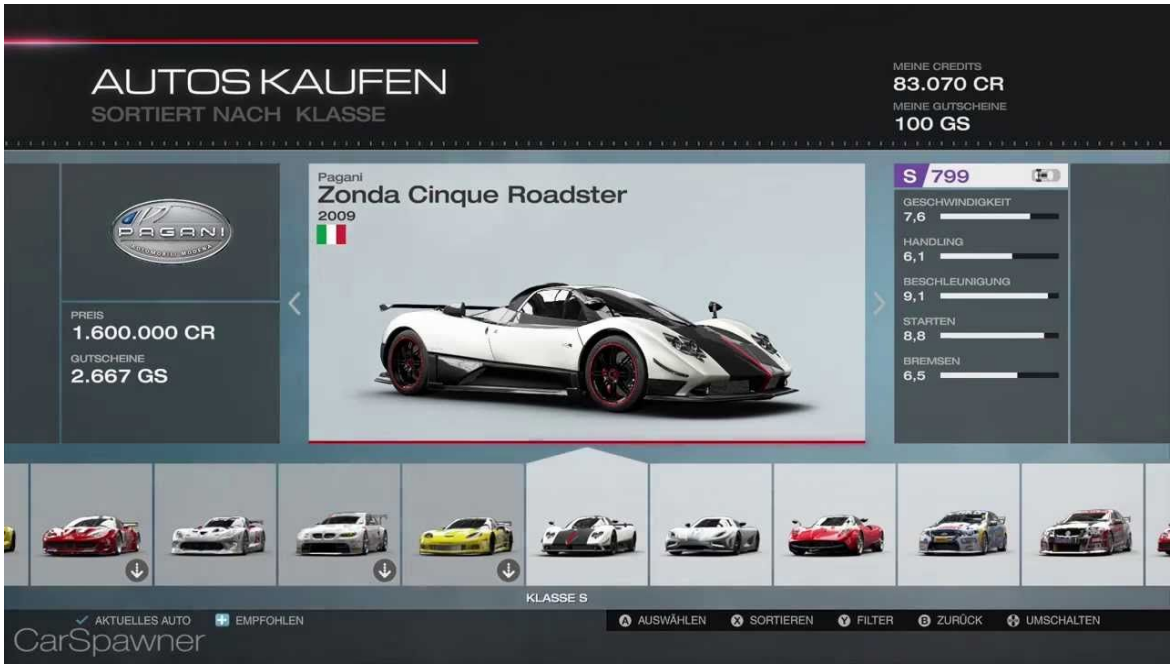
1) Calculadora. Se necesita validar la sintaxis de las ecuaciones ingresadas. Para ello, se debe analizar el equilibrio de los signos agrupadores (), [] y {}. El algoritmo debe agregar a la estructura de datos cada vez que encuentre el símbolo abierto, es decir: (, [o { y evaluar si el siguiente símbolo es el mismo pero cerrado, o sea:),] o }, por ejemplo:

Correcto: ( )(( )){([( )])}
Correcto: ((( )(( )){([( )])}))
Incorrecto:)(( )){([( )])}
Incorrecto: ({[ ])}
Incorrecto: (

Se pide:
Crear la función ***evaluar()*** que reciba como parámetro un array de caracteres e itere sus elementos evaluando si existe alguna inconsistencia. Retornará true si es correcto. Probar con:

entrada1 = {'a','+','b','*','-','[','(','-1','*','c',')','/','7',']','*','d'};
entrada2 = {'3a','+','{','-5x','-a','+','(','9x','-a','-x',')',']','}'};

2) Videojuego. Se debe crear la estructura de datos que dé soporte a un selector de autos de un videojuego simulador de carreras. Cada nodo deberá contener una estructura datos con los siguientes atributos: nombre, ruta a la imagen (cadena de caracteres), velocidad (entero), aceleración (decimal), consumo (decimal), peso (decimal) y coeficiente aerodinámico (decimal).



Se pide: El objetivo es definir una estructura que permita recorrer los nodos en ambos sentidos mediante la función ***mostrar()***, que recibirá por parámetro un puntero a la función que determine el sentido (del primero a último o al inverso), además del puntero a la lista de autos. Además se deberá implementar la función ***insertar()*** que insertará un nuevo nodo con datos de un auto a continuación de un puntero dado. El nodo actual (Pagani Zonda), se obtiene invocando la función ***irA(saltos-respecto-del-primero)*** que retorna un puntero al nodo seleccionado como se muestra en la figura. Implementar las funciones de soporte necesarias, por ejemplo: ***tamanio()***, ***vacia()***, ***inicio()***, ***ultimo()***, ***obtener(nombre)***.

3) Sensores IoT[1]. Se debe crear un módulo que recupere la secuencia de mediciones obtenidas desde un conjunto de sensores de temperatura ubicados en posiciones remotas y cuyos enlaces de comunicaciones son inestables.

Se pide:
Crear las estructuras necesarias.
Crear la función ***leer()*** que lea el archivo de texto donde se guardaron las mediciones (a medida que pudieron ser transmitidas) y agregarlas a una estructura de datos en memoria. Para una misma hora puede haber ninguna o muchas mediciones, por lo que las mediciones de una misma hora deberán ser parte de una sublista correspondiente a esa hora.
Cada nodo de la lista deberá contener el dato de la hora (entero) y un puntero otra lista cuyos nodos deberán contener: identificador del sensor (entero) y medición (decimal).

Los datos leídos del archivo de texto poseen el siguiente formato:
<id-sensor>, <hora:minutos-lectura>, <medición> (A razón de una medición por línea).

Mostrar por pantalla el contenido de cada nodo/subnodos con el siguiente formato:
<hora0>: <id-sensor-1, medición1>, <id-sensor-2, medición2>, <id-sensor-N, mediciónN>
<hora1>: <id-sensor-1, medición1>, <id-sensor-2, medición2>, <id-sensor-N, mediciónN>

Nota: El archivo de mediciones es suministrado. Para dividir por el caracter ',' las palabras de la línea con las mediciones, se recomienda emplear la función **strtok** (ver pág. 101 sección 1.3.3 del Anexo 1: https://droscarbruno.files.wordpress.com/2014/08/materialoficialayed_20141.pdf).

4) Antivirus. El objetivo es crear el módulo actualizador de firmas desde un archivo. Para ello se debe leer el archivo binario firmas.dat (proporcionado) que contiene los siguientes datos:

a. id (entero)
b. nombre (cadena de 25 caracteres)
c. firma (cadena de 25 caracteres)

El algoritmo que detecta virus analiza los archivos del sistema de archivos (file system del sistema operativo) y los compara con la firma de todos los nodos una lista almacenada en memoria.

Se pide:
Crear la estructura para leer el archivo de registros.
Crear una lista enlazada con los datos leídos del archivo. La función ***leer()*** deberá agregar a una lista en memoria nodos ordenados alfabéticamente por el campo Firma.
Crear el prototipo de la función ***analizar()*** que recibe como parámetro un array de cadenas de caracteres (archivos) y un puntero a la lista de firmas. Retorna un booleano indicando fue detectado un virus.
Crear la función ***guardar()*** que guardará el archivo firmas.dat con el contenido de la lista actualizada. Recibirá como parámetros un puntero a la lista de firmas y el nombre del archivo a guardar.

[1] http://internetofthingsagenda.techtarget.com/definition/smart-sensor