Algoritmos y Estructuras de Datos II

Recuperatorio Tema A - Tren de Carga

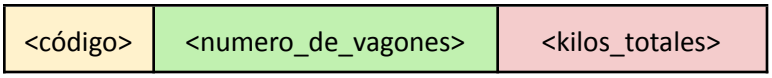
Ejercicio 1: Implementación de Tren de Carga

En el directorio del ejercicio se encuentran los siguientes archivos:

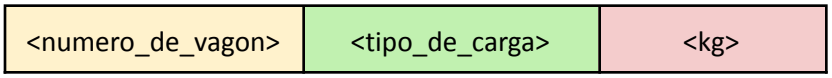
Archivo	***Descripción***
`main.c`	Contiene la función principal del programa
`wagon.h`	Declaraciones relativas a la estructura de los datos de vagones de tren y de funciones de carga y escritura de datos.
`wagon.c`	Implementaciones **incompletas** de las funciones
`array_helpers.h`	Declaraciones / prototipos de las funciones que manejan los datos del tren
`array_helpers.c`	Implementaciones **incompletas** de las funciones que manejan el arreglo

Abrir el archivo **`input/example1.in`** para ver cómo se estructuran los datos.

La primera línea contiene los datos del tren: código, cantidad de vagones, y peso total:



Luego le siguen una cantidad <numero_de_vagones> de líneas (una línea por cada vagón), para las cuales cada una contiene los datos de un vagón del tren.. El primer dato corresponde al número de vagón, luego sigue el tipo de carga del mismo, que puede ser: *rice*, *mushrooms*, *oatmeal* o *pepper*. El último dato es la cantidad de kilos de carga contenidos.. El esquema es el siguiente:



Consideraciones:

- Los vagones siempre tienen solo un tipo de carga, no hay restricciones sobre qué combinación de cargas puede llevar el mismo tren.
- Los vagones están numerados de 1 a *<numero_de_vagones>*

El ejercicio consiste en completar el procedimiento de carga de datos en los archivos **`array_helpers.c`** y **`wagon.c`**. Recordar que el programa tiene que ser *robusto*, es decir, debe tener un comportamiento bien definido para los casos en que la entrada no tenga el formato esperado. En particular se espera que si el número de kilos totales especificado en la primera línea del archivo no condice con la sumatoria de los vagones el programa reporte un error y termine.

Una vez completada la lectura de datos se puede verificar si la carga funciona compilando,

```
$ make
```

y luego ejecutar

```
$ ./train input/example1.in
```

Ejercicio 2: Análisis de los datos

Completar la siguiente función, definida en **`array_helpers`**

```
unsigned int discarded_wagons(Train t, unsigned int size);
```

Esta función debe retornar la cantidad de vagones en los cuales la carga debe ser descartada:

- Cada vez que hay más de dos vagones **contiguos** de carga *oatmeal,* todos los vagones intermedios entre estos (también cargados con *oatmeal*), deben ser descartados por contaminación con polillas. O sea si tenemos un grupo de 3 vagones con *oatmeal* tenemos que descartar el del medio.
- Tener en cuenta que no importa considerar los vagones *entre* dos vagones con *oatmeal.* La manera más fácil de calcular la cantidad de vagones descartados es sumar cada grupo de vagones contiguos de más de dos vagones de esta carga, y restar los dos iniciales por cada grupo.
    - 🌕 Lo cual es lo mismo que sumar cuántos vagones de *oatmeal* existen luego de dos vagones seguidos de *oatmeal.*

Consideraciones:

- Se provee el archivo **`Makefile`** para facilitar la compilación.
- Se recomienda usar las herramientas **`valgrind`** y **`gdb`**. ☣️
- Si el programa no compila, no se aprueba el parcial.
- Entregar código muy improlijo puede restar puntos
- **No modificar los array_helpers.h ni el main.c** ☣️