Córdoba 27 de Abril del 2021

Primer Parcial de Laboratorio

Algoritmos y Estructura de Datos II

TEMA A

Ejercicio 3

Ahora se trabajará sobre una estructura **`song_t`** definida como

```
typedef struct s_song_t {
  char song_name[MAX_NAME_LENGTH + 1u];
  char artist_name[MAX_ARTIST_LENGTH + 1u];
  unsigned int year;
  unsigned int seconds;
} song_t;
```

a) Completar en **`sort.c`** la definición de la funcion

```
bool goes_before(song_t s1, song_t s2)
```

de tal manera que devuelva `true` si y sólo si la cantidad de segundos de duración de `s1` es menor o igual a la duración en segundos de `s2`.

b) Hacer una implementación de `is_odd_sorted()` que trabaje sobre un arreglo con elementos del tipo `song_t`, es decir que tenga el siguiente prototipo:

```
bool array_is_odd_sorted(song_t playlist[], unsigned int length)
```

Debe basarse en el criterio de orden impuesto por `goes_before()`

c) Modificar **`main.c`** para que se muestre un mensaje indicando si la *playlist* está ordenada, usando `array_is_sorted()`, y para que muestre si está imparmente ordenada, usando `array_is_odd_sorted()`.

Para verificar pueden usar las *playlist* que les incluimos, debiendo obtener los siguiente resultados:

Playlist	***sorted***	***oddly_sorted***
`unsorted_joplin.lst`	**`false`**	**`false`**
`oddly_arg_rock.lst`	**`false`**	**`true`**
`sorted_queen.lst`	**`true`**	**`true`**