UNIVERSIDADE DA CORUÑA

FACULTADE DE INFORMÁTICA

Programación II – Curso 2022/23

Práctica 1: Enunciado

1. El Problema

El problema a resolver en esta Práctica 1 consiste en la implementación de un sistema de recuento de voto que se utilizará por los distintos jurados durante el próximo Festival de Eurovisión. El sistema llevará cuenta de los diferentes participantes que han recibido votos por un determinado jurado, el número de votos que tiene cada uno, la cantidad de votos nulos y la participación; además se permitirá descalificar participantes.

Como el objetivo de este trabajo es practicar la independencia de la implementación en los Tipos Abstractos de Datos (TADs), se pide crear dos implementaciones de una LISTA NO ORDENADA, las cuales deberán funcionar de manera intercambiable: una implementación ESTÁTICA y otra DINÁMICA. De este modo el programa principal no deberá realizar ninguna suposición sobre la forma en que está implementado el TAD.

Para facilitar la elaboración de esta primera práctica, se recomienda organizar el trabajo siguiendo las fases que se detallan a continuación.

2. Fase 1

Esta primera fase se centrará en el TDA. Para ello: (1) implementaremos una librería donde se incluyen los tipos de datos necesarios para el problema a resolver; y (2) implementaremos el TDA Lista en sus dos versiones, estática y dinámica.

2.1. Librería Types

Se definirán en esta librería (`types.h`) los siguientes tipos de datos, ya que son necesarios para el problema a resolver y los usará tanto el TAD como el programa principal.

`NAME_LENGTH_LIMIT`	Longitud máxima de un nombre (constante).
`tParticipantName`	Nombre del participante (`string`).
`tNumVotes`	Número de votos recibidos (`int`).
`tEUParticipant`	Pertenencia a la Unión Europea (`bool`).
`tItemL`	Datos de un participante. Compuesto por: • `participantName`: de tipo `tParticipantName` • `numVotes`: de tipo `tNumVotes` • `EUParticipant`: de tipo `tEUParticipant`

2.2. TAD Lista

Para mantener la lista de participantes y su información asociada, el sistema utilizará un TAD Lista. Se realizarán dos implementaciones:

1. **ESTÁTICA** con arrays (`static_list.c`) con tamaño máximo 25.
2. **DINÁMICA**, simplemente enlazada, con punteros (`dynamic_list.c`).

2.2.1. Tipos de datos incluidos en el TAD Lista

`tList`	Representa una lista de participantes.
`tPosL`	Posición de un elemento de la lista.
`LNULL`	Constante usada para indicar posiciones nulas.

2.2.2. Operaciones incluidas en el TAD Lista

Una precondición común para todas estas operaciones (salvo `createEmptyList`) es que la lista debe estar previamente inicializada:

- `createEmptyList (tList) → tList`
  Crea una lista vacía.
  PostCD: La lista queda inicializada y no contiene elementos.

- `isEmptyList (tList) → bool`
  Determina si la lista está vacía.

- `first (tList) → tPosL`
  Devuelve la posición del primer elemento de la lista.
  PreCD: La lista no está vacía.

- `last (tList) → tPosL`
  Devuelve la posición del último elemento de la lista.
  PreCD: La lista no está vacía.

- `next (tPosL, tList) → tPosL`
  Devuelve la posición en la lista del elemento siguiente al de la posición indicada (o `LNULL` si la posición no tiene siguiente).
  PreCD: La posición indicada es una posición válida en la lista.

- `previous (tPosL, tList) → tPosL`
  Devuelve la posición en la lista del elemento anterior al de la posición indicada (o `LNULL` si la posición no tiene anterior).
  PreCD: La posición indicada es una posición válida en la lista.

- `insertItem (tItemL, tPosL, tList) → tList, bool`

Inserta un elemento en la lista antes de la posición indicada. Si la posición es `LNULL`, entonces se añade al final. **Devuelve un valor `true` si el elemento fue insertado; `false` en caso contrario.**
PreCD: La posición indicada es una posición válida en la lista o bien nula (`LNULL`).
PostCD: Las posiciones de los elementos de la lista posteriores a la del elemento insertado pueden haber variado.

- `deleteAtPosition (tPosL, tList) → tList`
  Elimina de la lista el elemento que ocupa la posición indicada.
  PreCD: La posición indicada es una posición válida en la lista.
  PostCD: Las posiciones de los elementos de la lista posteriores a la de la posición eliminada pueden haber variado.

- `getItem (tPosL, tList) → tItemL`
  Devuelve el contenido del elemento de la lista que ocupa la posición indicada.
  PreCD: La posición indicada es una posición válida en la lista.

- `updateItem (tItemL, tPosL, tList) → tList`
  Modifica el contenido del elemento situado en la posición indicada.
  PreCD: La posición indicada es una posición válida en la lista.
  PostCD: El orden de los elementos de la lista no se ve modificado.

- `findItem (tParticipantName, tList) → tPosL`
  Devuelve la posición **del primer elemento de la lista** cuyo nombre de participante se corresponda con el indicado (o `LNULL` si no existe tal elemento).

2.2.3. Testeo de la implementación del TDA

Una vez terminada la implementación del TDA Lista, es necesario comprobar el correcto funcionamiento de la misma mediante el fichero de prueba facilitado (`test_list.c`).

3. Fase 2

Una vez implementado el TDA, nos centraremos en el programa principal. La tarea consiste ahora en implementar un único programa (`main.c`) que procese las peticiones recibidas por el sistema de votación del Festival, que tienen el siguiente formato:

`N participantName EUParticipant`	***[N]ew***: Alta de un participante con su número de votos a 0. `EUParticipant` puede tomar el valor *eu* o *non-eu*.
`V participantName`	***[V]ote***: Suma un voto al participante indicado.
`D participantName`	***[D]isqualify:*** Descalifica a un participante eliminándolo de la lista, y pasando a ser votos nulos todos sus votos.
`S totalVoters`	***[S]tats:*** Estadísticas de participación y de votos de cada uno de los participantes. `totalVoters` representa el número total de miembros de los diferentes jurados del certamen.

En el programa principal se implementará un bucle que procese una a una las operaciones. Para simplificar tanto el desarrollo como las pruebas, el programa no necesitará recibir ningún dato por teclado, sino que leerá y procesará las operaciones contenidas en un fichero (ver documento `Ejecucion_Script.pdf`). En cada iteración del bucle, el programa leerá del fichero una nueva operación y la procesará. Asimismo, para facilitar la corrección de la práctica todas las operaciones del fichero van numeradas correlativamente.

Para cada línea del fichero de entrada, el programa:

1. **Muestra una cabecera con la operación a realizar.** Esta cabecera está formada por una primera línea con 20 asteriscos y una segunda línea que indica la operación tal y como se muestra a continuación:

```
********************
CC␣T:␣participant/totalvoters␣XX␣location␣YY
```

donde `CC` es el número de petición, `T` es el tipo de operación (`N`, `V`, `D` o `S`), `XX` es o bien el nombre del participante (`participantName`) o bien el número de miembros del jurado (`totalVoters`) (según corresponda), `YY` indica si el participante representa a un país de la Unión Europea (*eu* o *non-eu* según el campo `EUParticipant`) y ␣ indica un espacio en blanco. Nótese que sólo se imprimirán los parámetros necesarios; por ejemplo, para una petición `[S]tats` se mostraría "`01 S: totalvoters 145`", mientras que para una petición `[N]ew` se mostraría "`02 N: participant P1 location eu`".

2. **Procesa la petición** correspondiente:

- Si la operación es **`[N]ew`**, se incorporará el participante **al final** de la lista de participantes con su número de votos inicializado a 0. Además, se imprimirá el mensaje:

  ```
  *␣New:␣participant␣XX␣location␣YY
  ```

  donde, nuevamente, `XX` es el `participantName`, `YY` es el valor `EUParticipant` (se imprimirá *eu* o *non-eu* en función del *bool*) y ␣ representa un espacio en blanco. El resto de mensajes siguen el mismo formato.

  Si ya existiese un participante con ese *participantName* o no se ha podido realizar la inserción se imprimirá el siguiente mensaje:

  ```
  +␣Error:␣New␣not␣possible
  ```

  No es posible incorporar participantes (petición `N`) una vez que la votación ha comenzado (petición `V`). Todos los ficheros de prueba proporcionados, así como los que se usarán en la corrección, cumplen esta premisa, por tanto, no es necesario comprobarlo.

- Si la operación es **[V]ote**, se buscará el participante, se incrementará su contador de votos en 1 y se mostrará el siguiente mensaje:

  ```
  *␣Vote:␣participant␣XX␣location␣YY␣numvotes␣ZZ
  ```

  Si no existiese ningún participante con ese *participantName* o si la lista está vacía, se imprimirá el mensaje:

  ```
  +␣Error:␣Vote␣not␣possible.␣Participant␣XX␣not␣found.␣NULLVOTE
  ```

  y este voto se contabilizará como voto NULO. Obsérvese que no hay un participante para representar este voto NULO, pero es necesario contabilizar estos votos.

- Si la operación es **[D]isqualify**, se buscará al participante en la lista, se borrará de la lista, pasando sus votos a considerarse como NULO y se imprimirá el siguiente mensaje:

  ```
  *␣Disqualify:␣participant␣XX␣location␣YY
  ```

  Si no existiese el participante `XX` en la lista o si la lista está vacía se imprimirá el siguiente mensaje:

  ```
  +␣Error:␣Disqualify␣not␣possible
  ```

- Si la operación es **[S]tats**, se mostrará entonces la lista completa de participantes, su porcentaje de votos y la participación actual de la siguiente forma:

  ```
  Participant␣XX1␣location␣LL1␣numvotes␣YY1␣(ZZ1%)
  Participant␣XX2␣location␣LL2␣numvotes␣YY2␣(ZZ2%)
          …
  Participant␣XXn␣location␣LLn␣numvotes␣YYn␣(ZZn%)
  Null votes NN
  Participation:␣KK␣votes␣from␣VV␣voters␣(PP%)
  ```

  donde `XX` es el `participantName`, `LL` indica si representa o no a un país de la UE (se imprimirá *eu* o *non-eu* en función del campo `EUParticipant`), `YY` es el número de votos recibidos por ese participante, `ZZ` es el porcentaje de votos recibidos por ese participante, `NN` es el número de votos nulos, `KK` es el número total de votantes (miembros del jurado) que han votado, `VV` es el número total de votantes existentes (`totalVoters`) y `PP` es el porcentaje de participación (con dos decimales).

  Los porcentajes de votos recibidos por cada participante se calculan sobre el total de voto válido (i.e. sin incluir nulos). El voto nulo no mostrará el porcentaje (sólo número) dado que no cuenta como voto válido. Si no se han recibido votos válidos, se indicará, para cada participante, un porcentaje de 0.00%.

  El dato de participación se calcula dando la proporción de miembros que han ejercido su derecho a voto (votos participantes + votos nulos) sobre el total de miembros de los jurados, el cual se adjunta en la propia orden `[S]tats`. Si no se han recibido votos, se indicará un porcentaje de 0.00%.

Si la lista está vacía se imprimirá el siguiente mensaje:

```
+ Error: Stats not possible
```

5. Ejecución de la práctica

Para facilitar el desarrollo de la práctica se proporciona el siguiente material de especial interés: (1) Un directorio `CLion` que incluye un proyecto plantilla (`P1.zip`) junto con un fichero que explica cómo hacer uso del mismo (Presentacion_uso_IDE.pdf) y, (2) Un directorio `script` donde se proporciona un fichero (`script.sh`) que permite probar de manera conjunta los distintos archivos proporcionados. Además, se facilita un documento de ayuda para su ejecución (`Ejecucion_Script.pdf`). Nótese que, para que dicho *script* de pruebas no dé problemas, se recomienda **NO copiar directamente el código de este documento**, ya que el formato PDF puede incluir caracteres invisibles que darían por incorrectas salidas (aparentemente) válidas.

6. Información importante

El documento `NormasEntrega.pdf`, disponible en la página web de la asignatura detalla claramente las normas de entrega. Para un adecuado **seguimiento de la práctica** se realizarán **entregas obligatorias parciales** antes de las fechas y con los contenidos que se indican a continuación:

1. Seguimiento 1: viernes **3 de marzo** a las 22:00 horas. Implementación estática y prueba del TAD Lista: entrega de los ficheros `types.h`, `static_list.c` y `static_list.h` (solamente dichos ficheros).
2. Seguimiento 2: viernes **10 de marzo** a las 22:00 horas. Implementación dinámica y prueba del TAD Lista: entrega de los ficheros `types.h`, `dynamic_list.c` y `dynamic_list.h` (solamente dichos ficheros).

Para comprobar el correcto funcionamiento de los TAD se facilita el fichero de prueba `test_list.c`. Se realizará una corrección automática usando el script proporcionado para ver si se supera o no el seguimiento (véase el documento `CriteriosEvauacion.pdf`).

Fecha límite de entrega de la práctica: jueves **16 de marzo** a las 22:00 horas.