Guía de ejercicios

INF2223 – Estructura de datos

ESCUELA DE INGENIERÍA INFORMÁTICA — PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

1. Enunciado

Debido a la reanudación de los eventos en el mundo producto de la flexibilización de las restricciones COVID-19, se ha producido un aumento considerable en la cantidad de eventos musicales y deportivos. Para ello, se contará con un sistema de gestión de eventos:

- **Sistema Eventos**: Contiene una referencia a una lista de artistas y la referencia a la cabecera de una lista simplemente enlazada circular que mantiene a los países.
- **ListaArtistas**: Contiene un contador de la cantidad de artistas que almacena y una referencia a la raíz de un árbol binario de búsqueda que gestiona a los artistas, El ABB se ordenan por el id.
- **Artista**: Cada artista tiene id (único en todo el sistema), nombre, género musical y el nombre del sello discográfico al cual pertenece.
- **País**: Tiene su nombre y una referencia a una lista doblemente enlazada con nodo fantasma que mantiene a los recintos que están en el país.
- **Recinto**: Tiene un id único en todo el sistema, nombre del recinto, su capacidad (tamaño máximo del arreglo) y un arreglo de referencias a eventos, el cual se encuentra compacto usando pLibre y ordenado por id del evento.
- **Evento**: Contiene un id de evento en todo el sistema, fecha del evento y una referencia al artista que se presenta.

```
struct SistemaEventos{
    struct ListaArtistas *artistas;
    struct NodoPais *paises;
};

struct ListaArtistas{
    int cantArtistas;
    struct NodoArtista *abbArtistas;
};

struct NodoArtista{
    struct Artista *datosArtista;
    struct NodoArtista *izq, *der;
};


struct NodoPais{
    struct Pais *pais;
    struct NodoPais *sig;
};
```

```
struct Pais{
    char *nombre;
    struct NodoRecinto *recintos;
};


struct NodoRecinto{
    struct Recinto *datosRecinto;
    struct NodoRecinto *ant, *sig;
};


struct Recinto{
    int id;
    char *nombre;
    int capacidad;
    struct Evento **eventos;
    int pLibreEventos;
};
```

```
struct Evento{
    int id;
    char *fecha;
    struct Artista *artista;
};

struct Artista{
    int id;
    char *nombre;
    char *genero;
    char *selloDiscografico;
};


#define tamArreglos 5000
```

2. Preguntas

Se pide implementar las siguientes funciones:

a. **`int agregarArtista(struct SistemaEventos *SE, struct Artista *nuevoArtista)`**: Agrega el artista recibido por parámetro al sistema. Debe validar que el artista no se repita. Retorna 1 en caso de éxito o 0 en caso de fracaso.

b. **`int agendarEvento(struct SistemaEventos *SE, int idEvento, id recinto, char *fechaEvento, int idArtista)`**: Recibe por parámetro el sistema de eventos, el id del recinto donde se debe agregar y los datos del evento a agendar. Debe crear el evento y agregarlo al recinto correspondiente. Debe validar que el recinto le quede espacio disponible y tenga libre la fecha indicada. En caso del artista debe validar que no tenga ocupada la fecha. En caso de éxito retorna 1 y en caso de fracaso 0.

c. **`int reprogramarEvento(struct SistemaEventos *SE, int idEvento, char *nuevaFecha)`**: Recibe por parámetro el sistema de eventos, el id del evento a ser reprogramado. Debe validar que el recinto tenga libre la nueva fecha indicada. En caso del artista debe validar que no tenga ocupada la fecha nueva. En caso de éxito retorna 1 y en caso de fracaso 0.

d. **`struct Evento *getEventosPorArtista(struct SistemaEventos *SE, int idArtista):`** Recibe por parámetro el sistema de eventos, el id del artista a consultar. Debe retornar una lista simplemente enlazada de los eventos que el artista tiene programados. Debe reutilizar los tipos de struct definidos en la parte superior (no puede crear nuevos tipos de structs).

No debe copiar datos, solo manejar referencias hacia los eventos existentes. Retorna el puntero a la lista o null en caso que el artista no exista o no tenga eventos programados.

e. **`int getIdRecintoMasOcupado(struct SistemaEventos *SE):`** Recibe por parámetro el sistema de eventos y entrega el id del recinto que tiene más eventos programados. En caso de emparte retorna el primero. En caso de fracaso retorna -1.

f. **`struct Pais **paisesConMasEventos(struct SistemaEventos *SE):`** Recibe por parámetro el sistema de eventos y entrega un arreglo de tamaño exacto de punteros a los países que tienen la mayor cantidad de eventos programados. En caso de que sea 1 solo país, el arreglo será de tamaño 1. En caso de fracaso retorna null.

g. **`struct Pais *paisConMasVariedadArtistas(struct SistemaEventos *SE):`** Recibe por parámetro el sistema de eventos y retorna un puntero al país que tiene la mayor variedad de artistas programados. Para ello, debe considerar que un artista puede tener programado varios eventos en el país, e incluso, en recintos distintos, en dicho caso cuenta sólo 1 vez. En caso de empate retorna al primer país. En caso de fracaso retorna null.

h. **`int getIdArtistaEnMasPaises(struct SistemaEventos *SE):`** Recibe por parámetro el sistema de eventos y debe retornar el id del artista que visita más países. En caso de empate retorna el primero. En caso de fracaso retorna -1.

i. **`struct Artista *quitarArtista(struct SistemaEventos *SE, int idArtista):`** Recibe por parámetro el sistema de eventos y debe quitar el artista del sistema. Para ello, debe eliminar todos los eventos programados del artista. Debe retornar una referencia al artista quitado. En caso de fracaso retorna null.

j. **`struct ListaArtistas *getArtistasSelloDiscografico (struct SistemaEventos *SE, char *selloDiscografico)`**: Recibe como argumento una referencia al Sistema Eventos y un nombre de sello discográfico. Debe retornar una Lista de Artistas que cumplan con la condición de que el artista pertenezca a dicho sello. En el caso de que no existan artistas o no se pueda efectuar la operación deberá retornar una Lista con null en su interior.

Importante:

- Procure modularizar las soluciones e implementar lo solicitado.
- Comente los supuestos
- Respete las reglas de programación y desarrollo.