Córdoba, 7 de Julio del 2023

Examen Final Regulares

Algoritmos y Estructuras de Datos II - Taller

El ejercicio consiste en implementar el TAD *PlayerOnB* que representa el estado de un jugador en el tablero de catan simplificado. En el juego “Los colonos de catán” cada participante puede tener sobre el tablero caminos, pueblos y ciudades. Si bien en el juego original la estructura es de tipo “grafo”, acá consideraremos una lista enlazada ya que contendrá los caminos, pueblos y ciudades del jugador como si fuera lineal. Cada nodo de la lista será de tipo:

- Camino (Road)
- Pueblo (Town)
- Ciudad (City)

Vamos a implementar al jugador en tablero utilizando una **lista enlazada de elementos** donde los nodos alojarán valores del tipo enumerado `type_t`:

```
typedef enum {Road, Town, City} type_t;
```

Las funciones a implementar en **`playerOnB.c`** son las siguientes:

Función	Descripción
`PlayerOnB playerOnB_empty();`	Devuelve un nuevo 'playerOnB' creado vacío
`PlayerOnB playerOnB_add_Road(PlayerOnB pob);`	Agrega un nuevo camino (Road) por izquierda
`PlayerOnB playerOnB_add_Town(PlayerOnB pob);`	Agrega un nuevo Pueblo (Town) por izquierda
`PlayerOnB playerOnB_upgrade_town(PlayerOnB pob, Position pos);`	Convierte el pueblo en la posición pos en una ciudad.SI LA POSICIÓN NO es válida, el método no hace nada. SI en la POSICIÓN no hay un pueblo, el método no hace nada. **IMPORTANTE: ESTE MÉTODO NO TIENE PERMITIDO MODIFICAR EL DATO DEL NODO.**
`bool playerOnB_is_valid(PlayerOnB pob);`	Devuelve True si el player on board es válido. Es válido cuando las ciudades o pueblos están separados por al menos 2 caminos.
`unsigned int playerOnB_get_score(PlayerOnB pob);`	Devuelve el puntaje del jugador en tablero. El puntaje se mide contabilizando 1 punto por cada pueblo y 2 puntos por cada ciudad, los caminos no suman puntos **IMPORTANTE: ESTE METODO DEBE SER CONSTANTE**
`unsigned int playerOnB_size(PlayerOnB pob);`	Devuelve la cantidad de elementos que tiene la lista de elementos del jugador en tablero (player on board) **IMPORTANTE: ESTE METODO DEBE SER CONSTANTE**

`void playerOnB_print(PlayerOnB pob);`	Imprime en pantalla el jugador en tablero. **IMPORTANTE: Es un método solo de lectura, no debe modificar la estructura**
`type_t *playerOnB_array(PlayerOnB pob);`	Devuelve un arreglo que representa el player on Board Este método debe pedir memoria para player on board y la memoria pedida debe ser liberada después
`PlayerOnB playerOnB_destroy(PlayerOnB pob);`	Destruye el jugador en tablero en cuestión y libera todos los recursos pedidos.

Puntaje

La función "`playerOnB_get_score`" devuelve el puntaje del jugador en tablero. Para ello cuenta la cantidad de pueblos y ciudades presentes. Sumará un punto por cada pueblo y un punto por cada ciudad. Los caminos no suman puntos. **ESTE MÉTODO DEBE SER CONSTANTE!**

Jugada correcta

En `playerOnB_is_valid` se debe chequear que la jugada en tablero sea válida. Para que sea considerada válida cada pueblo o ciudad deberá estar a una distancia >= 2 caminos entre sí. A continuación damos algunos ejemplos abreviando Road-> R, Town -> T y City -> C

Cerco	*Retorno*	*Razón*
`[T T C]`	`false`	es una jugada **incorrecto** porque no tiene caminos entre los pueblos
`[]`	`true`	**correcto**, No tiene ningún camino, pueblo ni ciudad
`[R R R R R]`	`true`	**correcto**, Sólo compuesto por caminos
`[R R T R R R R C T]`	`false`	**incorrecto**, hay un pueblo con una ciudad que no tienen caminos entre medio
`[T R R T R T]`	`false`	**incorrecto**, hay 2 pueblos separados por un solo camino
`[R R T R R R R C]`	`true`	**correcto**

Arreglos

La función `playerOnB_array()` debe devolver un arreglo dinámico con los elementos del jugador en tablero(del tipo `type_t`) en el orden apropiado (ver los ejemplos!). La cantidad de elementos contenidos en el arreglo se debe corresponder con el valor devuleto por `playerOnB_size()`.

Ejercicio 1: Implementar las funciones del TAD PlayerOnB y la estructura PlayerOnB

Ejercicio2: main.c

En el archivo main.c se encuentra el cuerpo de 3 test cases para ser completados. Completar con 3 casos de prueba que verifiquen el funcionamiento del TAD con distintos ejemplos. Se deben probar en estos casos el uso de todas las funciones definidas por el TAD

Consideraciones:

- El programa no debe contener *memory leaks*, bajan puntos
- Las funciones solicitadas que piden que sean de orden constante DEBEN SER DE ORDEN CONSTANTE
- Si el código no compila usando los *flags* del `Makefile`, el examen se considerará NO APROBADO (Asegurarse que compile antes de entregar)