Programación 2 - Primer recuperatorio

Dadas las siguientes clases, programar los métodos según el criterio planteado, usando únicamente metodología TDD:

| Evento |
| --- |
| - nombre: std::string  - fecha: unsigned int  - stands: Expositor \* [100] |
| + Evento (unNombre: std::string, unaFecha: unsigned int)  + ~Evento () + getNombre (): std::string  + getFecha (): unsigned int  + setFecha (unaFecha: unsigned int)  + inscribirExpositor(unNombre: std::string, unRubro: rubro\_t): Expositor \*  + desinscribirExpositor(unExpositor: Expositor &): bool  + operator[] (numeroStand: unsigned int): Expositor \*  + cantidadExpositores (): unsigned int  + cantidadRubros (): unsigned int |

Detalle:

* propiedad "fecha": fecha en formato AAAAMMDD.
* propiedad "stands": es un arreglo dinámico de punteros a objetos *Expositor* (definido abajo). Cada posición de este arreglo apuntará inicialmente a **nullptr** y se irá modificando en la medida en que se inscriban o desinscriban objetos *Expositor*. El "numeroStand" refiere al valor de subíndice + 1 (por ej., numeroStand=1 se ubica en stands[0]).
* método "operator[]": recibe un número de stand y retorna un puntero al objeto *Expositor* que se halla en dicha posición o **nullptr** si no estuviera asignada.
* método "inscribirExpositor": debe crear un objeto *Expositor* y ubicarlo en el primer lugar de *stands* que esté libre (**nullptr**). Luego retornará el puntero a dicho *Expositor* creado o **nullptr** si ya no quedaran lugares en *stands*.
* método "desinscribirExpositor": recibe una referencia a *Expositor* y debe liberar dicho lugar de *stands*. Retornará verdadero cuando la desinscripción sea exitosa y falso en caso contrario (por ejemplo, si *Expositor* no está en *stands*).
* método "cantidadExpositores": retorna cantidad total de objetos *Expositor* en *stands*.
* método "cantidadRubros": retorna la cantidad de rubros diferentes que hay en los objetos *Expositor* de *stands*. Es decir, si hay 2 objetos *Expositor* con el mismo rubro, deben contarse 1 vez.

| <<enumeration>>  **rubro\_t** |
| --- |
| BAZAR  GASTRONOMIA  FERRETERIA  INDUMENTARIA  LIBRERIA  OTROS |

| Expositor |
| --- |
| - nombre: std::string  - rubro: rubro\_t |
| + Expositor (unNombre: std::string, unRubro: rubro\_t)  + getNombre(): std::string  + getRubro(): rubro\_t  + setRubro(unRubro: rubro\_t) |

Detalle:

* propiedad "nombre": no puede ser un string vacío. Si fuera ese el valor establecido, deberá guardarse "Expositor indefinido" como nombre.

Criterios de corrección y aprobación:

* Se verá que los test desarrollados sean acordes al ejercicio planteado, con un nivel de exactitud razonable.
  + Considerar un universo amplio de posibilidades.
  + Testear éxitos y fracasos.
* La corrección consistirá en la ejecución de los tests que defina el docente. Por esta razón, es absolutamente necesario respetar rigurosamente los nombres de métodos, clases, propiedades, etc. dados ya que, de no hacerlo, los tests fallarán y restará puntos del examen.
* Pasar los tests del alumno y del docente es condición necesaria pero no suficiente para aprobar.
* Composición de la calificación:
  + Aplicación correcta de la metodología TDD: 50%
  + Implementación correcta del código aplicando los conceptos vistos en clase: 50%
  + Para aprobar el examen se requerirá, al menos, un 60% del total del examen.
  + La nota surgirá de la evaluación integral de la correcta aplicación de los temas abordados.

| Se provee un proyecto CodeBlocks pre armado. Al finalizar, eliminar las carpetas "bin" y "obj" del proyecto, comprimirlo en un archivo nombrado con tu apellido y subirlo a la tarea online correspondiente.  Link del proyecto:  <https://drive.google.com/file/d/1lqr4ikxuk4mMe5Nicf5XqaDUQcRjAC8B> |
| --- |

IMPORTANTE: Si no te permite subir el archivo, significa que tu Google Drive no tiene más espacio libre. Es necesario que hagas un poco de espacio para no tener este problema.