Instrucciones:

- Fecha de publicación: 23 de abril de 2020 a las 9:00.
- Fecha de entrega: 30 de abril de 2020 hasta las 23:55.
- Medio de entrega: `https://e-aulas.urosario.edu.co` (no se reciben entregas por correo electrónico u otros medios).
- La actividad **debe** realizarse **en grupos de tres** estudiantes.
- Formato de entrega: implementación, interfaz y driver en C++14.
- Nombre archivos: definidos más abajo.
- Importante: no use acentos ni deje espacios en los nombres de los archivos que cree.
- Solamente un miembro del grupo debe realizar la entrega en formato `zip`. Liste los miembros del mismo como un comentario en el encabezado de la **implementación**.

Protocolo para la evaluación:

Los siguientes lineamientos serán seguidos de forma estricta y sin excepción.

1. Los grupos pueden consultar sus ideas con los profesores para recibir orientación; sin embargo, la solución y detalles del ejercicio deben realizarlos **los integrantes de cada grupo**. Cualquier tipo de fraude o plagio es causa de anulación directa de la evaluación y correspondiente proceso disciplinario.
2. El grupo de trabajo debe indicar en su entrega de la solución a la actividad cualquier asistencia que haya recibido.
3. El grupo no debe consultar ninguna solución a la actividad que no sea la suya.
4. El grupo no debe intentar ocultar ningún código que no sea propio en la solución a la actividad (a excepción del que se encuentra en las plantillas).
5. Todas las entregas están sujetas a herramientas automatizadas de detección de plagio en códigos.
6. `e-aulas` se cerrará a la hora en punto acordada para el final de la evaluación. La solución de la actividad debe ser subida antes de esta hora. El material entregado a través de e-aulas será calificado tal como está. Si ningún tipo de material es entregado por este medio, la nota de la evaluación será 0.0.

No habrán excepciones a estas reglas.

Enunciado:

Resuelva el siguiente ejercicio sobre árboles binarios de búsqueda aumentados. No olvide compilar usando el estándar C++14, usando `-std=c++14`, junto con los *flags* para detectar *warnings* y errores: `-Wall -Wextra -Werror`.

Escriba su código en archivos de implementación plantilla (`dup_bst.cpp`), implementación que se debe entregar (`dup_bst.hpp`) y un archivo principal (`main.cpp`). Asegúrese de seguir cuidadosamente las indicaciones del ejercicio.

1. [*Binary search tree with duplicates.*] Una empresa tiene una base de datos confidencial que requiere ser organizada y codificada de forma tal que cierta información no puede ser fácilmente encontrada. La reorganización de la base de datos será realizada de acuerdo a una sola llave (*key*) y la información confidencial será codificada de tal forma que ciertos registros tendrán la misma llave. La idea del grupo de desarrollo de software de la empresa es organizar y codificar la información usando un árbol binario de búsqueda.

   Debido a que ciertos registros tendrán la misma llave, la implementación del árbol debe ser tal que este debe manejar duplicados. Por ejemplo, si las llaves que guarda el árbol son números enteros, al insertar un nuevo número que ya existe en el árbol, los nodos deberán mantener registro de cuántos enteros iguales han sido insertados anteriormente. Para lograr esto, los nodos del árbol deben ser aumentados; es decir, tener un atributo extra que da cuenta de cuántas veces se encuentra una llave (registro) en el árbol (base de datos). La estructura aumentada de un nodo del árbol binario de búsqueda es, entonces,

   ```
   template <typename keyType>
   struct BSTNode {
       int reps;       // duplicates of key
       keyType key;    // key of BSTNode
       BSTNode<keyType> *left;
       BSTNode<keyType> *right;
       BSTNode<keyType> *parent;
   };
   ```

   Esta aumentación de la estructura de un nodo impacta la implementación de las operaciones básicas de un árbol binario de búsqueda; en particular, las operaciones del problema del diccionario: búsqueda, inserción y borrado. El árbol binario de búsqueda aumentado también deberá implementar funcionalidad adicional que tiene que ver con el hecho que ahora se manejan duplicados.

   Como parte del grupo de software de la empresa, su tarea es la implementación de un árbol binario de búsqueda tal que ahora soporte llaves duplicadas. Específicamente, modifique o implemente, según corresponda, los siguientes métodos:

   a) `int insert(keyType k)`: método que inserta en el árbol la llave `k`, sin importar si está repetida, y retorna el número de repeticiones de la llave incluida la inserción actual.

b) `BSTNode<T>* find(keyType k)`: método que retorna un puntero al nodo que contiene la llave `k`, si esta se encuentra en el árbol; en caso contrario debe retorna `nullptr`.

c) `BSTNode<T>* remove(keyType k)`: método que elimina la llave `k` del árbol. Si la llave está repetida, eliminar significa disminuir `reps` en uno; si la llave no está repetida, entonces el nodo debe ser completamente eliminado usando `delete`.

d) `void display()`: método que imprime en orden el contenido del árbol. Este contenido debe ser mostrado en la forma de un par ordenado `(key, reps)`, con la intención de saber cuántas llaves están repetidas en el árbol.

e) `int capacity()`: método que calcula el número total de llaves registradas en el árbol. Es decir, calcula el número total de llaves como $\sum_{n=\texttt{nodes}} \texttt{reps}_n$, donde la suma se hace sobre el número total de nodos del árbol: internos y hojas.

Observe que la modificación de los métodos deberá tener lugar tanto en la sección privada como pública de la clase que implementa el árbol. Por último, asegúrese de revisar la correctitud de su implementación probando varios ejemplos para un árbol binario de búsqueda aumentado que guarda enteros y para uno que guarda `string`s.