OBJECT ORIENTED PROGRAMMING

Laboratory 13

OBJECTIVES

In this laboratory we'll work with smart pointers and we'll implement the simplest design pattern (or anti-pattern?).

LABORATORY

1. Implement the Singleton design pattern. Use a *unique_ptr* to store single instance of the class.

2. Write a *Date* class that has the instance variables *year*, *month,* and *day*. The requirement is that you should have one and only one object for the same date. For example, if in your application you need several date objects for *12 June 2020*, you need to create a single object with *year = 2020, month = 6* and *day* = 12 and use only this object throughout the application. You should also prevent copy construction and assignment for the Date class.

3. *** (This problem is mandatory if you did both the extra credit problems and you won't take the final lab exam)**
   Implement a class for a binary tree (you should use smart pointers for this, more specifically *shared_ptr;* it will all make sense next time).

   The nodes stored (*Node*) in the tree have the following data: pointers to the left and right children, a character *symbol,* and an unsigned integer that stores the *frequency* of the symbol.

   You class should have:

   - a parametrized constructor;
   - overload for operator<< ;
   - a static method which joins two nodes. The join operations takes as parameters two *Node* pointers: *left* and *right*, and returns another *Node* pointer for which the left child is the parameter *left,* the right child is the parameter *right,* the frequency is the sum of its left child's frequency and its right child's frequency, and the node's symbol is '*'.

   The *BinaryTree* class should store the root of the tree (a Node) as class member and should implement have functions to display the pre-order (root comes first **PRE**: **root**, then left subtree, then right subtree), the in-order (the normal order: left subtree, **root,** right subtree), and the post-order traversals. (root is last – **POST:** left subtree, right subtree, **root).**