# A Retail Stock Keeping System

## Due: 23:59 Friday, March 18, 2016

In this assignment you will implement a StockSystem class modeling a simple retail storefront inventory. The data storage of the StockSystem class will be supported by a RedBlackTree class.

Please review the general submission and coding style guidelines on the course website.

## Red-Black Tree Description

A documented declaration of a `Node` template class and `RedBlackTree` class is provided in redblacktree.h. Please refer to the comments in the header file for the definition and functional requirements. A subset of the `RedBlackTree` functions has already been implemented for you in rbtreepartial.cpp. It is highly recommended that you study the completed functions to become familiar with their operation. You are required to complete the implementation of the remaining functions in redblacktree.cpp:

- `CopyTree`
- `RemoveAll (private helper function)`
- `RBDeleteFixUp`
- `CalculateHeight`
- `constructors, destructor, operator=`
- `Insert`
- `Remove`
- `RemoveAll`
- `Size`
- `Height`

Please note that this implementation of Red-Black Tree is not to store duplicate values.

## `StockSystem` class

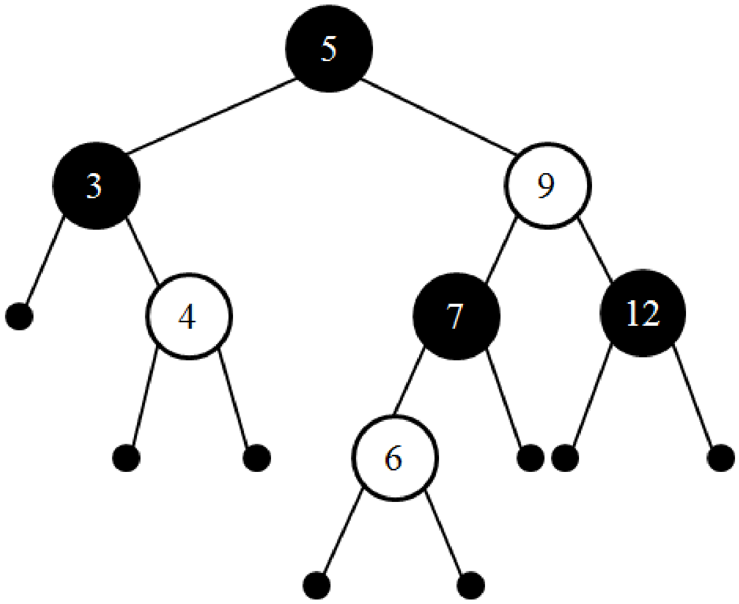
The `StockSystem` class supports some simple functions for managing a retail store. A `RedBlackTree` object is used to store records of the stock catalogue. The `StockSystem` class is to achieve its operations using calls to `RedBlackTree` methods only. Please refer to stocksystem.h for the class definition and functional requirements.

## Testing Tips

The `StockSystem` uses only a subset of the available `RedBlackTree` functions. Both classes will be thoroughly tested during grading – it is highly recommended to write a custom test driver or add your own code to the provided test driver to invoke all class functions and all their special and general cases.

If you wish to test the structure of your Red-Black Tree, you may make use of the `GetRoot()` function. Performing a pre-order traversal starting from the root node and displaying both the node contents and node colour will allow you to analyse the structure of your tree.

e.g. For the following tree,



A pre-order traversal which outputs the node contents and colour may return the sequence:

5*    3*    4    9    7*    6    12*

Performing insertion of the above sequence into an ordinary BST will replicate the structure of the original Red-Black Tree. You should write your own pre-order traversal in the testing driver which will output this structural information to you for verifying the structure of your tree.

## Deliverables

Include the following deliverables in your submission as a ZIP archive titled assign-4.zip submitted to CourSys:

- Title page listing the information of all contributing group members
- redblacktree.cpp
- stocksystem.cpp