CSC 212 Homework # 4
Recursion & Binary Trees
Due date: 05/04/2016

29/03/2016

Guidelines:	This is an individual assignment. The homework must be **submitted electronically through LMS**. **Hard copy submissions are not accepted**.

Problem 1

Write the **recursive** method *void removeEle(Stack <T >st, T e)* that deletes **all the occurrences** of the element *e* from the stack *st* keeping all other elements in their order.

Problem 2

Write the method *getEven* as user of the ADT binary tree. The method accepts a binary tree of integers and returns all the even numbers found in the tree as a list. The problem must be solved recursively. The method signature is *public List<Integer>getEven(BT<Integer>bt)*. Do not use any additional data structures.

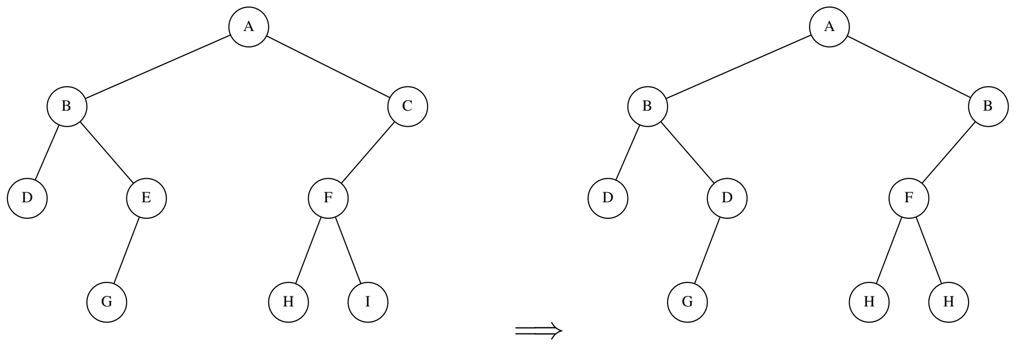
Problem 3

Write the method *getLeaf*, member of the class *BT* (binary tree) that accepts a node *t* and returns all of leaves' data in the subtree *t* using a list to store the results. The method must be written recursively. The method signature is *private List<T>getLeaf (BTNode<T>t)*.

Problem 4

Write the **recursive** method *private void copyLR(BTNode<T>t)*, member of the class *BT* (binary tree), that copies the data of the left child to the right child of every node in the subtree rooted at *t*.

Example 4.1. *See an example of the result of* copyLR *in the figure below.*



Problem 5

An arithmetic expression can be represented as a binary tree as shown in the figure below.

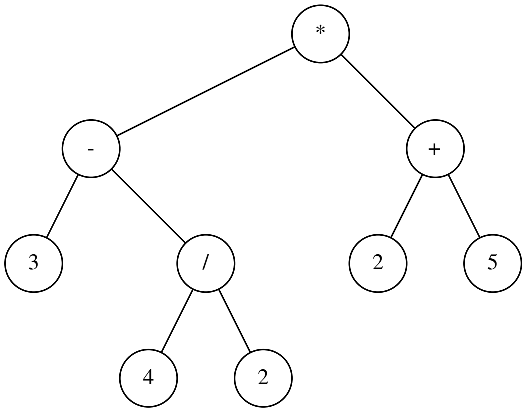


Figure 1: Representation of $(3 - 4/2) \times (2 + 5)$

Write the **recursive** method *public double eval(BT<Token>expr)* that evaluates the expression *expr*. The interface *Token* represents a token of the expression and is described as follows.

```
public enum TokenType {
        Operand,
        Operation
}
```

```
public interface Token {
        public TokenType getType(); // Returns the type of the
            token
        public double getVal(); // Returns the value of the token
            if it is of type Operand
        public double apply(double op1, double op2); // If the
            token is of type Operation, this method applies it to
            the operands passed as parameters and returns the
            result of the operation.
}
```

Assume that the tree *expr* is not empty. Use the methods of the class *BT* to navigate and access the tree.