CSC 212 Homework # 3 - II
ADT Stack
Due date: 08/11/2015

01/11/2015

Guidelines:	This is an individual assignment. The number of "*" indicates the difficulty of the question. Fill in this page, **print it out and use it as a cover page** to your homework. Write your name, student ID number, section number and homework number on each page as a header. Staple your homework. The parts designated **online** must be submitted electronically to Web-CAT. The homework must be handed in hard copy to your tutorial instructor.
Name:	....
Student ID:	....
Section:	....

Problem 1

1. Evaluate the following postfix expression using a stack. Redraw the stack **after every push** operation: 5 5 6 + 3 2 / $\times$

2. Solve the following infix expression using stacks. Redraw the stacks **after every push**: 6+1$\times$5-7>2

. . . .

Problem 2

1. Write the method *pushBack* (user of the Stack ADT), that takes a stack *st* and an integer *p*, and pushes *the top of the stack* to the p^{th} position. The top element in the stack is in position 1. Assume that $1 \leq p \leq n$, where n is the length of *st*. The method signature is *public <T>void pushBack(Stack<T>st, int p)*. **Do not use any additional data structures.**

   Example 2.1. *Assuming the stack st (from top to bottom):* $1, 2, 5, 3, 10, 6$. *Calling* s.pushBack(*st*, 4) *will result in st* : $2, 5, 3, 1, 10, 6$.

2. As a user of the ADT Stack, write the following method that checks if the top element of a stack of Integers is equal to the total sum of all lower elements in the stack. It returns true if they are equal and false otherwise. The stack should not be changed after you call the method. The method signature is *public boolean checkTotalTop(Stack<Integer>st)*.

. . . .

Problem 3 (*)

1. Write the method printAll, that prints all elements of a tree (**not necessarily binary**) starting by the root and following the order explained in the example below. The method signature is *public <T>void printAll(TreeNode<T>root)*. **Use only one stack.**

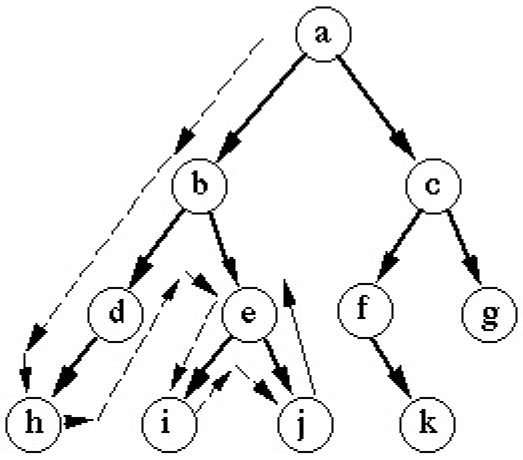

Figure 1: A tree

Example 3.1. *Calling* printAll(root)*, where root is the starting node of the tree, will result in printing* $a, b, d, h, e, i, j, c, f, k, g$.

Each node in the tree is of type TreeNode<T>described below. Do not complete the code, use only the methods provided in this class.

```
public class TreeNode<T> {
   public T data;
   .....

   public int nbChildlren(){
   // return the number of children of the node
   ...
   }

   public TreeNode<T> getChild(int k){
   // return the k-th child of the node
   ....
   }
}
```

2. Suppose you want to print the tree in the following order: $a, b, c, d, e, f, g, h, i, j, k$ (level by level). What data structure would you use in this case? Rewrite the method *printAll* for this order.

. . . .