

CSC 212 Homework # 3  
Stacks & Recursion  
**Due date: 03/12/2016 at 5:00 AM**  
**(Morning)**

---

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

---

**Problem 1**

1. Convert the following expression to postfix notation, show the stacks after each push:  $a - b * c / d * e ^ f + g$ .
2. Trace the evaluation of the following expression, show the stack after each push:  $6 5 2 ^ 2 3 + 8 * - 3 - *$ .
3. Convert the following expression to infix notation, show the stack after each push:  $6 5 2 ^ 2 3 + 8 * - 3 - *$ .
4. Trace the evaluation of the following expression, show the stacks after each push:  $5 + 6 ^ 2 / 2 / 3 - 2 * 4 * 7$ .

**Problem 2**

1. Write the static method *removeBottom* (user of Stack ADT) that takes a stack *st* as input, and removes the bottom element of *st*.

Signature: `public static<T> void removeLast(Stack<T> st)`.

**Example 2.1.** Assuming *st* (top-to-bottom): 5, 7, 5, 3, 2. After calling *removeBottom(st)* *st* will be: 5, 7, 5, 3.

2. Write the method *topEqualsBottom* that checks if the top element of the stack is equal to bottom element. Return true if that is the case. Method signature is `public static <T> boolean topEqualsBottom(Stack<T> st)`. The stack *st* should not change after the method has been called.

### Problem 3

1. Given an array of integers, write the **recursive** method *containsMult3* that checks whether the array contains a multiple of 3. You have to choose the appropriate signature for *containsMult3*.

**Example 3.1.** For the array `[1, 3, 2]`, the method returns true, for `[1, 4, 8]`, it returns false, and for `[9]`, it return true.

2. Given an array of integers, write the **recursive** method *sameSign* that checks whether all the elements of the array have the same sign (all strictly positive or all strictly negative).

**Example 3.2.** For the array `[-1, -3, -2]`, the method returns true, for `[4, 0]`, it returns false, for `[2, -3, 9, 1]`, it return false, and for `[2, 3, 9, 2]`, it return true.

### Problem 4

1. Write the recursive method `public boolean recSearch(T k)` member of the class *LinkedList* that searches the list for element *k*. It should return "true" if found false otherwise. Do not use other data structures. You can add a private member method as needed.

**Remark.** Recursive member functions are private in general, since their parameters may depend on the internal representation of the data structure. Consequently, when one talks about a recursive public method, it is understood that the method itself is **not recursive**, but calls a private recursive method that does the job.

2. Write the **recursive** method `public void reverse()`, member of the class *ArrayStack* that reverses the content of the stack.

### Problem 5

1. Write **recursive** method *insertAtBottom* (user of the Stack ADT), that takes a stack *st* and an element *e*, and insert element *e* at the bottom of the stack. The method signature is `public <T> void InsertAtBottom(Stack<T> st, T e)`.

2. Write **recursive** method *reverse* (user of the Queue ADT), that takes a Queue *q* and reverses the order of the elements in the queue *Q*. The method signature is `public <T> void reverse(Queue<T> q)`.
3. Write a **recursive** method *merge*(*q*<sub>1</sub>, *q*<sub>2</sub>) that merges the queues *q*<sub>1</sub> and *q*<sub>2</sub> into a new queue. After the call, *q*<sub>1</sub> and *q*<sub>2</sub> become empty (**Do not use any loops**). Signature: `public <T> Queue<T> merge(Queue<T> q1, Queue<T> q2)`.

**Example 5.1.** *If the queue *q*<sub>1</sub> contains: A, B, C, and *q*<sub>2</sub> contains: D, E, F, G, H, then the result of merge(*q*<sub>1</sub>, *q*<sub>2</sub>) is: A, D, B, E, C, F, G, H.*

4. Rewrite the method *merge* so that the two queues *q*<sub>1</sub> and *q*<sub>2</sub> do not change after the call (**Do not use any loops**).