CS3323 Fall 2006 Assignment 2
Due Monday, Oct. 16, by 5pm.

- Assignments should be handed in by placing them in the CS3323 bin on E level of Gillin Hall.
- A hardcopy of your program code must be submitted with your written assignment, code must also be submitted electronically by email to hzhang@unb.ca as an attachment. The name of your attachment must consist of your last name, student id, assignment number, question number. For example, your last name is Smith, student id is: 330321, it is the code for Question 6 in Assignment 2, then the name of your attachment should be "Smith-330321-2-6.java".

---

1. Describe the output of the following series of stack operations on a single, initially empty stack:
   $push(5)$, $push(3)$, $pop()$, $push(2)$, $push(8)$, $pop()$, $pop()$, $push(9)$, $push(1)$, $pop()$, $push(7)$, $push(6)$, $pop()$, $pop()$, $push(4)$, $pop()$, $pop()$.

2. Describe the output of the following series of queue operations on a single, initially empty queue:
   $enqueue(5)$, $enqueue(3)$, $dequeue()$, $enqueue(2)$, $enqueue(8)$, $dequeue()$, $dequeue()$, $enqueue(9)$, $enqueue(1)$, $dequeue()$, $enqueue(7)$, $enqueue(6)$, $dequeue()$, $dequeue()$, $enqueue(4)$, $dequeue()$, $dequeue()$.

3. Describe in pseudo-code a linear-time algorithm for reversing a queue Q. To access the queue, you are only allowed to use the methods of queue ADT. *Hint*: Consider using an auxiliary data structure.

4. Describe how to implement two stacks using one array. The total number of elements in both stacks is limited by the array length; all stack operations should run in $O(1)$ time.

5. Describe in pseudo-code a linear time - $O(n)$ - algorithm which copies the elements of array A into a new array B in such a way that B contains all the elements of A with any odd integers located before any that are even. For example,

2 3 1 6 8 9 4 7 5 10 ← A

becomes something similar to:

3 1 9 7 5 2 6 8 4 10 ← B

Hint: The order of the integers in each grouping (odd/even) in B need not be preserved.

6. Write a Java program to calculate postfix expressions using a stack, and convert postfix to infix. The input to your program will be text, consisting of one expression per line. All expressions will consist of numbers and operators from the operator set {+, -, *, /}, separated by spaces. For each expression, compute both the infix equivalent of the expression and its value, and print them as an equation. It is acceptable to add parentheses even if they are not needed.

   If an input expression is not a correctly formed postfix expression (eg., 5+6), an error message should be displayed.

   To simplify reading the input, all numbers in the input expressions will be nonnegative integers. This will not necessarily be true of intermediate values of the output.

   Example input:
   22 6 5 + / 9 -
   10 7 - 17 8 1 + - *
   8 5 / 3 +
   5 + 6

   Example output:
   22 / (6 + 5) - 9 = -7
   (10 - 7) * (17 - (8 + 1)) = 24
   8 / 5 + 3 = 46
   Not a valid postfix expression.