Homework 2
COP 3530 Data Structures and Algorithms
Due date: 11:59pm, Feb. 3, 2014

Important Notes:

1. Please name your java class HW1_Last_4_digits_UIN. Do not write your name anywhere in your program.
2. You are given a testing driver. Please use it without changing anything.
3. When submitting, zip your source code and submit it on Canvas.

Assignments:

In this homework, you are supposed to develop and implement a ***stack-based*** and ***non-recursive*** algorithm that can convert a prefix to its infix. You need to consider parentheses in your program. In other words, you need to figure out when and where to insert parentheses in the infix.

Here are some hints for this assignment. Note: It will be difficult to understand these hints by just looking at the words. It is HIGHLY recommended that you manually simulate this process at least once.

1. You can scan the prefix from the left to the right and form the infix in the same way.
2. The stack will be only used for operators. Each operator will be pushed on the stack. It is important to figure out what needs to be done before pushing an operator.
3. When scanning a certain symbol, there are five cases: (In the following discussion, X, Y, and Z indicate operators and A, B, and C indicate operands.)
   - Operator X follows Operator Y
     - This implies that evaluating X results in the left operand for Y.
     - Where can we find Y? Because X follows Y, Y must have just been pushed on the stack; in other words Y is at the top of the stack.
     - If Y has the precedence over X, you need to append a left parenthesis to the output string and push a right parenthesis on the stack.
     - Pushing a right parenthesis on top of Y is to make sure the just appended left parenthesis is closed before popping Y.
     - At last, push X on the stack.
   - Operator X follows Operand A
     - This implies that evaluating X results in a right operand of another operator, say Y.
     - This also implies that the left part of operator Y has been (almost) accomplished.
     - Therefore, the next symbol in the infix should be operator Y. And Y should also be the top operator in the stack.
     - However, Y may be covered by one or more right parentheses. So you need to check to see if this is the case. If so, you need to pop the right parentheses and Y. Of course, they will be appended to the output.

    - If X does not have the precedence over Y, you still need to append a left parenthesis to the output string and push a right one on the stack.
    - At last, push X on the stack.
- Operand A follows Operator X
    - Please figure out how to deal with this case.
- Operand A follows Operand B
    - Please figure out how to deal with this case.
- Please find out what the last case is and how to deal with it.

At last, please use the examples in lab 1 for testing.

Use this main method

```
public static void main(String args[]){

        String inFix, preFix;
        System.out.println("Enter an infix string: ");
        inFix=readString();

        preFix=infix_to_prefix(inFix); // you have the code from lab 1
        inFix=prefix_to_infix(preFix);

        System.out.println("The prefix from infix is "+preFix);
        System.out.println("The infix from prefix is "+inFix);

}
```