CS526
Homework Assignment 5

Due: 10/21

There are two parts in this assignment. Part 1 is a practice of the design of an recursive algorithm and Part 2 is a practice of using stacks.

Part 1

Write a recursive java code that receives an array of integers and rearrange them in such a way all odd integers appear before all even integers. The order of integers is not important. Note that you may not use an additional array and the rearrangement of integers must occur within the given array *a*. Name it *HW5_part1.java*. An incomplete code is posted on Blackboard, which contains a main method, which you can use to test your recursive code.

This part is a practice of the **design** of a simple recursive algorithm. Once you have a design, implementing it will be straightforward.

Part 2

You are required to write a Java program that evaluates and produces the value of a given arithmetic expression using a stack data structure. For example, if the given expression is (2 + (5 * 3)), your program must produce 17. As another example, consider the expression ((4 – 2) * ((19 – 13) / 2)). Given this expression, your program must produce 6.

We make the following assumptions about the input expression:

- All operands in the input expressions are positive integers. But, the value of an expression could be a negative number.
- Only the following operators are allowed: +, –, *, and /
- Expression is fully parenthesized. In other words, each operator has a pair of "(" and ")" surrounding its two operands.

Your program must read a number of arithmetic expressions from an input file, named *infix_expressions.txt*, and produce an output. A sample input file is shown below:

```
( ( 2 + 5 ) * 3 )
( ( ( 3 + 2 ) * 3 ) – ( 6 / 2 ) )
( ( ( 7 – 5 ) * 2 ) / ( 10 – 8 ) )
```

Each line has one arithmetic expression. Operands, operators, and parentheses are separated by a space. So, you can use a space as a delimiter when tokenizing an expression. The expected output for the above input is:

```
The value of ( ( 2 + 5 ) * 3 ) is 21
```

The value of (((3 + 2) * 3) – (6 / 2)) is 12
The value of (((17 – 15) * 2) / (10 – 8)) is 2

Note that a given expression is repeated in the corresponding output. You must write the output on the screen.

A pseudocode is given below. You must implement this pseudocode.

```
Read an input expression (this is single line) from the input file and tokenize it.

Create two stacks – one for operands and the other for operators

Scan the tokens in the expression from left to right and perform the following:

•  If a token is an operand, push it to the operand stack.
•  If a token is an operator, push it to the operator stack.
•  If a token is a right parenthesis, pop two operands from the operand stack and pop an
   operator from the operator stack and apply it to the operands. The result is pushed back to
   the operand stack.
•  If a token is a left parenthesis, ignore it.
•  After all tokens are processed, what is left in the operand stack is the result.

Repeat the same process for all expressions in the input file.
```

We assume that all expressions in the input file are valid (i.e., you don't need to check for invalid expressions).

For the two stacks, you must use the ***LinkedStack.java*** class that is included in the textbook's source code collection. You may not use Java's stack class or a stack class defined by somebody else. Note that you must not modify the ***LinkedStack.java*** class. Name your program as *HW5_part2.java*.

Documentation

No separate documentation is needed. However, you must include sufficient inline comments within your program.

Deliverables

You need to submit the following files:

- *HW5_part1.java*
- *HW5_part2.java*
- All other necessary files, including:
    - *LinkedStack.java*
    - *Stack.java*
    - *SinglyLinkedList.java*

  - *Other files, if any*

Combine all files that are necessary to compile and run your programs into a single archive file. Name the archive file *LastName_FirstName_hw5.EXT*, where *EXT* is an appropriate file extension, such as *zip* or *rar*. Then, upload it to Blackboard.

Grading

Each part is worth 10 points.

Part 1 Grading:

- The TA will test your program with two integer arrays and up to 4 points will be deducted for each wrong output.
- Up to 2 points will be deducted if your program does not have sufficient inline comments.

Part 2 Grading:

- The TA will test your program with four expressions (that are in a single test input file) and up to 2 points will be deducted for each wrong output.
- Up to 2 points will be deducted if your program does not have sufficient inline comments.