605.202 Data Structures and Algorithms LAB 1  
Use of Stacks  
Use stacks to evaluate if a given string is in a language L. Your code will not be recursive. In your  
analysis, suggest a recursive algorithm and compare with your stack based solution. The  
purpose of using a stack is to take advantage of its LIFO nature, therefore algorithms which merely  
use the stack for storage and determine inclusion of the string in the language by the use of counting  
the input string in any manner will NOT receive any credit.  
Let L1= { w: w contains equal numbers of A's and B's (in any order) and no other characters}  
L2 = { w: w is of the form AnBn, for some n > 0 }  
L3 = { w: w is of the form AnB2n, for some n > 0 }  
L4 = { w: w is of the form (AnBm)p, for some m,n,p > 0 }  
L5 = { a non-trivial language of your choice}  
Examples of languages which are only trivially different from L1 and L2:  
L5 = { w: w contains equal numbers of C's and D's (in any order) and no other characters},  
L5 = { w: w is of the form CnDn for some n > 0 },  
L5 = { w: w is of the form BnAn for some n > 0 }.  
w = AAABBB  
AB  
ε ( the empty string)  
ABABABA  
ABAB  
BBAA  
BBBAA  
AAB  
AABBCCD  
ABCBA  
ABBBA  
ABBA  
ABAABBAAABBB  
AABACABAA  
AABBBAABBB  
Test each string given as well as additional strings you make up yourself against each of the five  
languages. A sixth language would be considered an enhancement.  
Be sure to discuss your data structures and their implementation and why they make sense. E.g.  
why is stack a reasonable choice to solve this problem? What implementation of a stack did you  
choose? Why? As stated above, consider a recursive solution and compare it to your iterative  
solution. Is one better than the other? If so, why?  
Note: You are expected to write the stack code yourself and not use the library stack class. Be sure  
to include the code for your stack as part of the source code you submit.

Lab1

How to run program:

1. Check program includes all tree java files: Lab1.java, LanguageL.jave, Stack.java.
2. Place input file (with testing strings) and output file (empty file) at designed location
3. Source code can be compiled with Java 1.8 from the command line
4. Execute “java Lab1[inputfile path][outputfile path]” at command line to kick off program