CMPS 12B-02, Fall 2017
HW3: Balanced (Multiple) Brackets

Due: Sunday, Nov 5, 2017 by 11:59pm

- All assignments must be submitted through git. Please look at the Piazza guide on submitting assignments.
- Please follow instructions properly, include naming conventions, and carefully read through the input/output formats. Please review the Piazza guide on the checking script. Run the checking script to make sure your files are named correctly. You may get partial credit (as described in the Grading section below) if the checking script fails in certain ways.
- If your HW3 solution is submitted by Sunday, November 5, 2017 by 11:59pm, then it may get full credit. One point will be deducted for late HW3 solutions that are submitted by Monday, November 6 by 11:59pm. Solutions submitted after that will not be graded. Don't forget to fill in your commit id in the Google Form.
- Clearly acknowledge sources in a README file, and indicate if you discussed the problems with other students or groups. Please submit a README file even if you didn't use any sources. In all cases, the course policy on collaboration applies, and you should refrain from getting direct answers from anybody or any source. If in doubt, please ask the Instructor or the TAs.

1 Problem description

Main objective: Determine whether any expression with multiple kinds of brackets, braces and parentheses is balanced using a reference-based stack. You must implement the stack directly yourselves. Using a stack library, or JCF stacks, or a list ADT will earn no credit.

Types of brackets: There are four "parenthesis/bracket" pairs:
- **(and)**
- **[and]**
- **{ and }**
- **< and >**

A line may contain other characters (including spaces) besides these 8 characters, but those characters are ignored. An expression is balanced if all the "open parentheses" are closed by a matching "close parentheses", and nested parenthesis match, so that (for example) you can't have "]" closing "{", or a stray extra ")" when there isn't an open "(".

For example, the following lines are all not valid (N):

ab(x]w
alpha{ beta<gamma}delta>
(([[Turing]])
The (quick [brown (fox (jumped) over) the] lazy) dog)
< { }
> html <

And the following lines are all valid (Y):

ab[x]w
alpha{ beta<gamma>delta}
([[Turing]])
The (quick [brown (fox (jumped) over) the] lazy) dog
metamorphosis
< html >
([< { } >]) < < ( ) > >

You will not get any credit if you do not implement your solution to HW3 using a reference-based stack. Furthermore, you have to write your own stack from scratch. You cannot use any built-in libraries for stacks or linked lists, nor can you use an ADT for lists or an array-based stack.

Format: You should provide a Makefile. Running `make` should create "Balanced.jar". Your program should take two command line arguments, an input file and an output file.

Input The input file consists of a series of lines as described above, each (including the last line) ending with an end-of-line.

Output: On running (say) :

```
java -jar Balanced.jar my-input.txt my-output.txt
```

the file `my-input.txt` is read in and the file `my-output.txt` should be output. The i'th line of the output file `my-output.txt` corresponds to the i'th line in the input file `my-input.txt`. The i'th line of the output file should contain just :a single character, as follows:

- If the i'th line is not valid, then the i'th line should just be "N" (without the quotes).
- If the i'th line is valid, then the he i'th line should just be "Y" (without the quotes).

Examples: Piazza has a zip file for HW3 called `HW3files.zip`. The files `test-input.txt` and `test-output.txt` (which should help you see if your program is correct) are for the checker. Two additional files, `more-input.txt` and `more-output.txt` that are in the zip file will also help you test your program.

2 Grading

You code should terminate within 3 minutes for all runs. If it doesn't, we will not give you credit. As usual, your submission must past the checker to get full credit; the spec file, HW3-spec.txt, is in the same zip file, `HW3files.zip`, that was mentioned above. To receive credit ,you must also submit your git commit id using the Google Form for HW3.

1. (10 points) Full solution as described above.
2. You will lose one point for each error that you have when we test your program on `test-input.txt` and on `more-input.txt`
3. You will receive 0 points if the checker script fails because you submitted the wrong files.