Assignment 4 – Stacks and Queues

Deadline: Thursday, February 28, 2025, 9:59 pm

Objectives:

- Familiarize yourself with the Stack and Queue data structures. You may use ONLY Stacks and Queues for this assignment.

Instructions:

For problems marked as **(code)** – You should write a Java program that will be run by the instructor
For problems marked as **(text)** – You should write an answer. The answer can be either prose, an algorithm (pseudo-code or code), a proof, or mathematical equations. Whichever form of text answer is appropriate to solve the problem. In case the answer is an algorithm (code or pseudo code), it will be assessed mostly on logic rather than on whether it compiles.

Problems:

1 (text) Stacking [10 points] Given an empty stack, what will be the contents of the stack after the following operations? (Hint: When you have a method inside another method call (e.g., this(that()), that() will be executed first, afterwards this() will be executed)

- push(8)
- push(2)
- pop()
- push(pop()*2)
- push(10)
- push(pop()/2)

2 (text) Queueing [10 points] Given an empty queue, what will be the contents of the stack after the following operations? (Hint: When you have a method inside another method call (e.g., this(that()), that() will be executed first, afterwards this() will be executed)

- push(4)
- push(pop()+4)
- push(8)
- push(pop()/2)
- pop()
- pop()

3 (text) Find in deque [10 points] We discussed that using a doubly-linked list, you are able to search an element in $O(\frac{n}{2})$, the same is true for a deque.

Given a Deque q and an element y**,** provide an algorithm that finds the position in the deque in which element x is stored in $O(\frac{n}{2})$.

Hint: The i-th element from the right it a position i, whereas the i-th position from the left it a position n-i

4 (code) Balanced Brackets [25 points]
A bracket is considered to be any one of the following characters: (,), {, }, [, or].
Two brackets are considered to be a *matched pair* if the an opening bracket (i.e., (, [, or {) occurs to the left of a closing bracket (i.e.,),], or }) *of the exact same type*. There are three types of matched pairs of brackets: [], {}, and ().

A matching pair of brackets is *not balanced* if the set of brackets it encloses are not matched. For example, {[(])} is not balanced because the contents in between { and } are not balanced. The pair of square brackets encloses a single, unbalanced opening bracket, (, and the pair of parentheses encloses a single, unbalanced closing square bracket,].

By this logic, we say a sequence of brackets is *balanced* if the following conditions are met:

- It contains no unmatched brackets
- The subset of brackets enclosed within the confines of a matched pair of brackets is also a matched pair of brackets

Given a String of brackets, determine whether the sequence of brackets is balanced. If a String is balanced, return YES. Otherwise, return NO.

Function Description: Write a function named *isBalanced*.
isBalanced has the following parameter(s):

- *String s*: a String of brackets

Returns

- *String:* either YES or NO

Input Format
A single String **s** will contain only a sequence of brackets. Can be empty.

Constraints

- $1 \leq |s| \leq 10^3$, where $|\boldsymbol{s}|$ is the length of the sequence
- All characters in the sequences ∈ { **'{'**, **'}'**, **'('**, **')'**, **'['**, **']'** }
- You are allowed to use the LinkedList class but only the add(), remove(), and peek() methods

Output Format
For each String, return **YES** or **NO**

Sample Inputs
{[()]}
{[(])}
{{[[(())]]}}

Sample Output
YES
NO
YES

Explanation

1. The String **{[()]}** meets both criteria for being a balanced String.
2. The String **{[(])}** is not balanced because the brackets enclosed by the matched pair **{** and **}** are not balanced: **[(]**.
3. The String **{{[[(())]]}}** meets both criteria for being a balanced String.

5 (code) Decode String [25 points]

Given an encoded string, return its decoded string.
The encoding rule is: k[encoded_string], where the encoded_string inside the square brackets is being repeated exactly k times. Note that k is guaranteed to be a positive integer.

You may assume that the input string is always valid; there are no extra white spaces, square brackets are well-formed, etc. Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repeat numbers, k. For example, there will not be input like 3a or 2[4].

The test cases are generated so that the length of the output will never exceed 105.

Example 1:
Input: s = "3[a]2[bc]"
Output: "aaabcbc"
Example 2:
Input: s = "3[a2[c]]"
Output: "accaccacc"
Example 3:
Input: s = "2[abc]3[cd]ef"
Output: "abcabccdcdcdef"

Constraints:

- 1 <= s.length <= 30
- s consists of lowercase English letters, digits, and square brackets '[]'.
- s is guaranteed to be a valid input.
- All the integers in s are in the range [1, 300].

6 (code) Infix to postfix [10 points]

Given a String input containing an infix expression. Return a String output containing the equivalent postfix expression.
Infix expression: The expression of the form a op b. When an operator is in-between every pair of operands.
Postfix expression: The expression of the form a b op. When an operator is followed for every pair of operands.

Note: The order of precedence is: ^ greater than * equals to / greater than + equals to -. Ignore the right associativity of ^.

Hint:

- If the scanned character is a **(** , push it to the stack.
- If the scanned character is a **)** , pop the stack and output it until a **(** is encountered, and discard both the parenthesis.

Constraints

- The input is guaranteed to be a valid expression.

Sample Input
a+b*(c^d-e)^(f+g*h)-i
Sample Output
abcd^e-fgh*+^*+i-

7 (text) Algorithm Analysis [10 points]

For each of the algorithms you wrote for problems 4-6, explain their time complexity and space complexity using Big-O notation. Explain how you arrived at your answer.

Grading Rubric:

Item	Points
Stacking	10
Queueing	10
Find in Deque	10
Balanced brackets	25
Encode String	25
Infix to Postfix	10
Algorithm Analysis	10
Total Points	100

Deliverables:

On Canvas, the link to a GitHub repository with (1) the implementation of the (code) problems done using an IntelliJIDEA project and (2) a pdf with the answers to the (text) problems. Make sure you double-check that your code and pdf file were uploaded correctly to GitHub and that the link was uploaded correctly on Canvas! I will not be accepting excuses after the deadline that the files were not uploaded correctly or that you forgot to submit.

You may submit a physical (by hand) solutions to (text) problems. Please be sure your work is written clearly and readable.

Compiling -- There will be an automatic 5% penalty for each compile error in your code that has to be fixed in the grading process -- up to a maximum of 10 compile errors. After 10 errors fixed, if it still fails compilation, the submission will be marked as a 0. (Bottom line: Make sure your code compiles before you submit it!!!)

Failure to submit all the required files in the appropriate format will result in a 50% penalty.