# Lab 9 Java Sorting  Regex

UWYO COSC 2030

## 1 Lab: Java Refresh on Stacks, Queues, Deques, and Vectors

Stacks, Queues, Deques, and Vectors are all data structures that you have used before in this class, though in different languages. Now you will be using them in Java to do tasks which should be familiar to you at this point: reversing strings, checking parentheses, and sorting. Links to the Java documentation for each has been included below:

- `https://docs.oracle.com/javase/8/docs/api/java/util/Stack.html`
- `https://docs.oracle.com/javase/8/docs/api/java/util/Queue.html`
- `https://docs.oracle.com/javase/8/docs/api/java/util/Vector.html`
- `https://docs.oracle.com/javase/8/docs/api/java/util/Deque.html`

Once you have completed these familiar tasks, you will then get started with regular expressions (regex) in Java.

Github Classroom Link: `https://classroom.github.com/a/o-_vze1f`

## 2 Lab: Java String Reversal and Parentheses Checking

Using the Java program Lab9.java on the website, complete the functions stringReverse and parenCheck. String reverse will use a stack, parenCheck will use a queue. You will then complete stringReverseVector and parenCheckDeque in which you will need to use a deque and a vector, respectively.

### 2.1 String Reversal

Stacks use the last in first out style for storing data. This means if you push in the characters 'h','e','l','l','o' into a stack that it will come out in the reverse order.

### 2.2 Parentheses Checking

For this you will check sets of parenthesis to ensure they are done properly. Each time you get a '(' push it on to the data structure. You will pop it when you get a matching ')'. For a perfectly matched string you should have an empty data structure by the end. What happens when you find a ')' with no matching '(' to pop?

## 3 Lab: Java Sorting

You will finish implementing the same basic sorting algorithms that you have seen before in previous labs. There will be three algorithms included in the lab Quicksort, Mergesort, and Heapsort. Quicksort will already be completed for you, it is there for you to reference. Mergesort and Heapsort will need to be completed.

### 3.1 Heapify and Mergesort

Complete the functions heapify and mergesort. DO NOT change main or the function declarations. Start with a MAXSIZE to be set to 100 (defined up top) to ensure you are sorting correctly, you can uncomment the print statements in main if it helps. Make sure you re-comment out the print statements after you ensure you are sorting correctly.

Once you are sorting correctly, increase MAXSIZE up to 10,000 and run it again. You will not need to time your code.

- Mergesort is missing the RECURSIVE calls
- Mergesort also needs to put the recursive calls together
- heapify also needs RECURSIVE calls

## 4 Java Regular Expressions

In this lab you will create three functions that use regular expressions to accept or reject a variety of strings. The match conditions for the three regexes are:

- 1. If the input string contains at least one occurrence of 'Cowboys'
- 2. If the input string is a 5-digit string starting with '7'
- 3. If the input contains any character other than 'z', '$', 'J', or '@'

A link to a reference on regex in Java has been included below:

- `https://www.w3schools.com/java/java_regex.asp`

## 5 Turn in on Github. Make sure you include a readme with your name and lab section.