**Assignment 3**

1. (50 %) Some programming languages do not allow you to implement queues, but they have already mechanisms to implement stacks. The goal of this exercise is for you to implement a queue using two stacks. There are two deliverables of this exercise:
2. Create a video of yourself explaining how this implementation of the queue works. I don’t want you to explain the Java code, but to provide through an animation, or a video of you explaining the implementation of the queue. Use your cell phone, or ask somebody to record you while you’re explaining. Upload your video to YouTube and provide a link to it (your CSU Gmail account allows you to have a YouTube account, you just need to activate it). Provide this link as a comment in the code when you submit your file. You can also create a power point presentation, record your presentation explaining how to do this. You can use the freely available software <https://obsproject.com/> to record your presentation and upload it to YouTube.
3. Provide an implementation of QueueADT presented in the book (slides 21-22 of slide14.pptx file). Submit your implementation on a java class/file called **Queue2Stack.java**. You can use the implementations of stacks provided by the Java API for this exercise (you don’t need to implement your own linked data structure to solve the problem).

2. (50 %) An electronic device store sells only Peach and Cyborg mobile devices and the way business is performed is on a strictly "first in, first out" (FIFO) basis. Customers must buy the "oldest" (based on arrival time) of all the mobile devices in the store, or they can select whether they would prefer a Peach or a Cyborg device (and will receive the oldest device of that type). They cannot select which specific device within each brand.

Create queue data structures under the class CellDeviceStore to maintain this system and implement operations such as enqueueDevice, dequeueAnyDevice, dequeuePeach and dequeueCyborg. The interface of this class is provided by the attached file interface Assigment3Store. Make your implementation of CellDeviceStore to implement this interface. Make the class that you create for this exercise to be part of the package **csu.datastructures.lab3** . You can use the implementations of queues provided by the Java API to solve this exercise (you don’t need to implement your own linked datastructure).

Assigment3Store interface makes use of a class called Device (provided). There’s a method on the Device class (isOlder) that returns true if the current object is older than the one passed as parameter to the method. To setup a device’s date you can use a call to Calendar.getInstance().getTimeinMillis() inside the enqueueDevice method (this method returns the current time in milliseconds) .

**Submit:**

Your implementation of CellDeviceStore class (the CellDeviceStore.java class)