Guidelines:

- **Make sure to submit your files to Google Classroom before the deadline,** otherwise, your work won't be considered for grading.
- **Submit only the java files as separate files (i.e. not as zipped files).**
- **Create a class for every program named based on the question name then write all the necessary methods and/or the main method**

Queue Implementation:

Write the class MUListQueue that is a single-linked list implementation of a queue.

1. Add the code of Node<E> class to the class. You can get this from MUSingleLinkedList class file.
2. Add all necessary data fields in the class.
3. Write the code of the methods *peek*, *poll*, and *offer* that are in Lesson 6 Slides.
4. Add a method **public void moveToRear()** that moves the element currently at the front of the queue to the rear of the queue. The element that was second in line will be the new front element. **Do not use *offer* and *poll* methods**.
5. Test your methods in a main method. You will need to implement toString method to print your queue. You can get its code from previous exercises.

QueuePractice:

Create a class QueuePractice that contains the following static methods:

1. **public static void reverse()**: the method reads in a sequence of integers and inserts them into a queue (-1 to stop). It then uses a stack to reverse the queue and prints it on the screen. Use only methods of the Queue interface

```
5
23
2
5
2
-1
[5, 23, 2, 5, 2]
[2, 5, 2, 23, 5]
```

2. **pubic static void removePrime()**: the method reads in a sequence of integers and inserts them into a queue. It then removes prime numbers from the queue. You can write a method that checks whether a number is prime or not. Use only methods of the Queue interface

```
7
5
1
2
3
-1
[7, 5, 1, 2, 3]
After removing prime numbers:
[]
```

```
7
13
4
5
6
1
0
2
-1
[7, 13, 4, 5, 6, 1, 0, 2]
After removing prime numbers:
[4, 6]
```

3. Test your methods in a main method.

DequePractice:

Create a class DequePractice that contains the following static methods:

1. **public static void numericOrNot()**: the method reads a sequence of integers and inserts each integer that is positive at the front of a deque and each integer that is negative at the rear of a deque (0 to stop). Your method should also count the number of strings of each kind. It then displays the message "Integers that are positive" followed by the positive integers and then the message "Integers that are negative" followed by the negative integers. Do not empty the deque. Use only methods of the Deque interface.

```
4
7
3
-5
3
-6
-3
11
0
Integers that are positive
0
11
3
3
7
Integers that are negative
-5
-6
-3
```

2. **public static void reverse()**: the method reads in a string of characters and stores each character of the string in a deque. Then, it displays the deque contents. Then it uses a second deque to store the characters in reverse order. When done, it displays the contents of both deques. Use only methods of the Deque interface.
3. Test your methods in a main method

```
Enter a string of characters
good morning!
Contents of deque
[g, o, o, d,  , m, o, r, n, i, n, g, !]
deque1  deque2
g       !
o       g
o       n
d       i
        n
m       r
o       o
r       m
n
i       d
n       o
g       o
!       g
```