METROPOLITAN STATE UNIVERSITY

ICS 240 - 02: Introduction to Data Structures
Fall 2018

Assignment 7: Queues, Sorting, and Recursion
Out: Friday, November, 16th, 2018
Due: Friday, December, 7th, 2018

Total points: 50

Implement the following methods.

Question 1 (7 Points): write a static method, called `replace`, that takes three parameters: a queue and two integers, `oldVal` and `newVal`. The method then changes the input queue by replacing each occurrence of `oldVal` with `newVal`.

Question 2 (8 Points): write a static method, called `split`, that takes as input parameter a queue, `inputQ`, that includes positive integers. The method returns two queues as output where the first queue includes all even values from `inputQ` and the other queue includes all odd values from `inputQ`. The input queue should be empty after calling this method.

Question 3 (10 Points): Implement the selection sort algorithm to sort integers that are stored in a linked list. You are given the class `IntNode` that implements a linked list node that stores integer data. The method header is as follows:

```
public static void selectionSort(IntNode head)
```

Start from the code that implements selection sort on an array and think how can you change that given the fact that, in a linked list, you cannot access elements by specific index. On the other hand, you have to keep a reference to the linked list nodes as you iterate over them. **Important:** You have to implement the algorithm **without** using any auxiliary arrays or linked lists. (e.g. you will get zero for this question is you copy all data in an array, sort the array, and then put the data again in the linked list).

Question 4 (6 Points): In this question, you are to rewrite the linked list selection sort method but using a generic linked list node `Node<E extends Comparable>`. You can then use this method to sort any collection of `Comparable` objects (objects that are stored in the linked must implement the `Comparable` interface so that they can be sorted). The method header is as follows:

```
public static <E extends Comparable> void selectionSort(Node<E> head)
```

Note that there are only few differences you need to make from the code that you wrote in Question 3. The more important thing is to make sure you understand the differences.

Question 5 (6 Points): write a **recursive** method, called `sumover`, that takes one integer parameter `n`, which is a non-negative integer. The method returns a double value, which is the sum of the reciprocals of the first n positive integers, where the reciprocal of x is the fraction 1/x. For example, `sumover(1)` returns 1.0 (which is 1/1); `sumover(2)` returns 1.5 (which is 1/1 + 1/2); and `sumover(3)` returns approximately 1.833 (which is 1/1 + 1/2 + 1/3). Define `sumover(0)` to be zero. Do not use any local variables in your method.

Question 6 (7 Points): The British have been shelling the Revolutionary forces with a large cannon from within their camp. You have been assigned a dangerous reconnaissance mission -- to infiltrate the enemy camp and determine the amount of ammunition available for that cannon. Fortunately for you, the British (being relatively neat and orderly) have stacked the cannonballs into a single pyramid-shaped stack. At the top is a single cannonball resting on a square of four cannonballs, which is itself resting on a square of nine cannonballs, which is itself resting on a square of sixteen cannonballs, and so forth. Given the danger in your situation, you only have a chance to count the number of layers before you escape back to your own encampment. Write a **recursive** method that take as input the number of layers and returns as output the number of cannonballs in the stack.

Question 7 (6 Points): Write a **recursive** method, called `printAstricks`, that takes two integer parameters, `m` and `n` such that `0 <= m` and `m <= n`. The method prints a line of `m` asterisks, then a line of `m+1` asterisks, and so on up to a line of `n` asterisks. Then the same pattern is repeated backward: a line of `n` asterisks, then `n-1`, and so on down to `m`. The only loop allowed in your implementation is a loop to print a line of `m` asterisks. You may have two copies of this loop in different places of the implementation. For example, `printAstricks(3,6)` print the following:

```
***
****
*****
******
******
*****
****
***
```

What to submit

You will submit only one Java file called `Driver.java` that includes the implementation of the 7 methods. However, note that your solutions for questions 3 and 4 will make use of the `IntNode` and `Node` classes. Also, to test the sort method in Question 4, you will need to use a class that implements the Comparable interface (e.g., `Circle`). I added on D2L a zip file called `'assignment7-code.zip'` that includes these three files (`Node.java`, `IntNode.java`, and `Cirlce.java`).