Central Washington University
College of the Sciences
Department of Computer Science

CS-301 Data Structures   Fall 2016

Lab Practice 09

This lab aims at practicing the implementation of operations in a variety of data structures, like stacks and queues. At the same time, since this is the last lab, we will review do's and dont's of interfaces, abstract classes, constructors, etc.

Reminder: as stated on the syllabus, it is very important that you complete the lab, within two or three days.

Normally you, will find the source and data files in `/home/cs-301/Labs/Lab09`.

```
ld09.pdf
CircArrayQ.java
UtilQueue.java
Problem.java
SinglyLinkedList.java
```

1. The java source file `UtilQueue.java`, although apparently syntax-error free, it has a number of problems. Modify the *bodies* of the methods described, so the class compiles correctly

2. Implement correctly the methods above in `UtilQueue.java.`, as described in the source program. Use a client to test your methods.

3. In class we began the implementation of a `queue` for integers using a circular array: `CircArrayQ.java` Complete the implementation, by providing a constructor for a given *capacity* as a parameter, and the following missing methods:

   ```
   public int front()   //     the front element
   public void dequeue()  //  removes front element
   public int size()    //    number of elements in the queue
   ```

   Hint: Remember that such implemention can only hold *one less* element than the capacity of the array.

4. Test your class with a suitable client, and verify that a small queue actually works as expected.

5. The stack ADS provides a natural structure to evaluate arithmetic expressions, as described in class. Write a simple calculator `Calc` for aritmetic expression in *postfix* Allow floating point numbers as operands. The expression is provided at command line, separated by spaces. From the example in class

   ```
   Calc  2 4 * 9 5 + -
   ```

   must display -6.0.

6. Some of the methods in `SinglyLinkedList<E>` were not implemented, either because they were optional in `AbstractList<E>` or because there was no immediate need. By looking at the source file and the comments, target some of the methods that would be nice to have and implement them. Test them appropriately. Rename your class.