Joanna Klukowska
joannakl@cs.nyu.edu

Homework Assignment 4

Problem 1

Given the IntegerQueue ADT below:

Informal Specification

The `IntegerQueue` contains a (possibly empty) collection of objects of type `Integer`. The queue supports the following operations:

insert	This operation adds a `Integer` object, given as a parameter, to the end of the queue of integers.
remove	This operation removes and returns a `Integer` object from the front of the queue of integers. If the queue is empty, `null` should be returned.
toString	This operation produces a String that contains all Integer objects stored in the queue from font to back separated by a single comma and a single space. If the queue is empty, an empty string should be returned.

State the return value and show the content of the, initially empty, queue of `Integer` objects after each of the following operations. If any of the operations below is invalid or would cause program to crash, state it and explain what is wrong with it (then proceed with the next step ignoring the invalid line). Assume that `queue` is a reference of type `IntegerQueue`.

```
queue.insert( 15 );
queue.insert( 3 );
queue.insert( -15 );
queue.insert( 35 );
queue.remove();
queue.remove();
queue.remove();
queue.insert( 13 );
queue.remove();
queue.remove();
queue.remove();
queue.insert( 3 );
```

Assume that the implementation is array based and follows the efficiency ideas that we discussed in class. Assume that the initial capacity of the array to store the queue is equal to 4 and that its size is doubled whenever the array runs out of room.

Problem 2

Given the CharStack ADT that we discussed in class:

Joanna Klukowska
joannakl@cs.nyu.edu

Informal Specification

The CharStack contains a (possibly empty) collection of objects of type Character. The stack supports the following operations:

insert / push	This operation adds a Character object, given as a parameter, to the top of the stack of characters.
remove / pop	This operation removes and returns a Character object from the top of the stack of characters.
peek	This operation returns a Character object from the top of the stack of characters.
toString	This operation produces a meaningful String representation of the stack.

Show the content of the, initially empty, stack of Character objects after each of the following operations. If any of the operations below is invalid or would cause program to crash, state it and explain what is wrong with it. Assume that `stack` is a reference of type `CharStack`.

```
stack.push('c');
stack.push( new Character('s') );
stack pop();
char p = 's';
stack.push( p );
stack.push( p );
stack.push( new Character('1') );
stack.peek();
stack.pop();
stack.push('%');
stack.peek();
stack.push('A');
stack.push('X');
stack.pop();
stack.pop();
```

Assume that the implementation is array based and follows the efficiency ideas that we discussed in class. Assume that the initial capacity of the array to store the stack is equal to 4 and that its size is doubled whenever the array runs out of room.

Problem 3

The `GenericList` interface and its `GenericLinkedList` implementation used in class (see the source code for lecture 5) provide an insertion method that always adds a new node to the back of the list. Assuming that the data item that is stored in the `GenericNode` object implements the `Comparable` interface (i.e. has standard definitions of compareTo(...), write the method

```
void orderedInsert( T item )
```

that adds a new node to the list in a sorted order (from smallest to largest).

Joanna Klukowska
joannakl@cs.nyu.edu

Problem 4

Answer the following true/false questions. You do not need to explain your answers, but you should make sure you understand why they are what they are.

T F A stack is a first in, first out structure.

T F A queue is a last in, first out structure.

T F A Java interface must have at least one defined constructor.

T F A class that implements an interface has to implement ALL methods listed in the interface.

T F In a non-empty stack, the item that has been in the stack the longest is at the bottom of the stack.

T F A recursive solution to a problem is always better than an iterative solution.

T F A general case of a valid recursive algorithm must eventually reduce to a base case.

T F Recursive methods should be used whenever execution speed is critical.

T F If you enqueue 5 elements into an empty queue, and then perform the dequeue operation 5 times, the queue will be empty again.

T F If you push 5 elements into an empty stack, and then perform the peek operation 5 times, the stack will be empty again.

T F In a singly linked list based implementation of a queue, at least one of the operations that add or remove an item has to traverse all the elements in the queue.

T F In a singly linked list based implementation of a stack, , at least one of the operations that add or remove an item has to traverse all the elements in the stack.

Problem 5

In doubly linked lists there are references to both the first and last elements of the list. The node used for such a list is defined as follows:

```
public class TwoWayNode <E> {

    private E data;
    private TwoWayNode<E> next;
    private TwoWayNode<E> previous;
    public TwoWayNode (E data ) {
        this.data = date;
        next = null;
        previous = null;
    }

}
```

Assume that the class provides all the getters and setters.

Implement

Joanna Klukowska
joannakl@cs.nyu.edu

```
void enqueue (E item)
```

and

```
E deque()
```

methods of a Queue <E> class that is based on such a doubly linked list. Your queue keeps private data fields called `head` and `tail` that are both reference variables of type `ToWayNode`<E>.

Make sure to document your code (tell the reader what you are doing).

Problem 6

Write a **recursive** mathod that computes the number of nodes in a singly linked list. Make sure you cover all the special cases. Your method is a public method of a LinkedList class. Assume that the `Node` class has a data field called `value` that stores the value of a node and a data field called `next` that references the next node if such exists or null if there is no next node. The LinkedList class has a data field called `head` that points to the first node or null if the list is empty. Use the following method header. You may use a helper method if you find that appropriate.

```
public int size ()
```

Problem 7

Determine if the following prefix + and postfix expressions are valid or not. Evaluate their value, if they are valid. All numbers are single digits.

```
a) 1 2 + 3 *
b) 1 2 3 + *
c) 1 2 + * 3
d) 1 + 2 * 3
e) + 1 * 2 3
f) + * 1 2 3
g) 3 2 1 + 5 * + 4 -
h) 3 + 2 1 * 5 + 4 -
i) - 3 + 2 * 1 + 5 4
j) - 3 + 2 * 1 5 + 4
```