CSC230 Fall 2014
Program #6

The purpose of this assignment is to practice:

- Manipulating doubly linked lists
- Working with generic types
- Working with Strings
- Using the merge algorithm

Use the DLinkedList class posted this on Blackboard. You MUST NOT CHANGE ANYTHING IN THE FILE. Add your code, but do not alter the existing code.

You will add the following methods to this file:

DLinkedList methods

- A **remove()** method that removes a single node containing the *value* passed in as a parameter from the list. Method signature is provided and may not be changed.
- An **insertOrder()** method that inserts a value into the list in such a way that the list remains sorted in ascending order. Method signature is provided and may not be changed.
- An **insertOrderUnique()** method that does the same as **insertOrder()** with the additional requirement that the value is only inserted in the list if it is unique. Method signature is provided and may not be changed.
- A **merge()** method that merges the calling list with the parameter list in such a way that the result is a list that is sorted in ascending order and contains no duplicate values. The resulting list should be returned. The two original lists should be empty when the function exits (not destroyed). Method signature is provided and may not be changed.

I am providing a main method. You will add a static method called **cleanUp()** that takes a String as an argument. The method should return the String after removing any leading or trailing non-alphabetic characters. The resulting String should be in all lower-case. Method signature is provided and may not be changed.

Ideally, **merge()** should only make one pass through each list. **merge()** is tricky to get right — it may be solved iteratively or recursively. There are many cases to deal with: either 'a' or 'b' may be empty, during processing either 'a' or 'b' may run out first, and finally there's the problem of starting the result list empty, and building it up while going through 'a' and 'b'. To get full credit – this method must not use **insertOrder or insertOrderUnique or remove**.

The main method reads from 2 files and builds the lists as it goes with “cleaned” words. It outputs each list, calls merge, and then outputs the 2 initial lists (which should be empty) and the resulting list.

Here’s a sample run for the following input files:

File1:
Baby Face!
You've got the cutest little Baby Face!
There's not another one could take your place
Baby Face

File2:
I'm bringing home a baby bumblebee,
Won't my mommy be so proud of me,
I'm bringing home a baby bumblebee,
Won't my mommy be so proud of me,
Ouch! It stung me! (spoken)

```
run:
Before merge()
lst1:  [another, baby, could, cutest, face, got, little, not, one,
place, take, the, there's, you've, your]
lst2:  [a, baby, be, bringing, bumblebee, home, i'm, it, me, mommy, my,
of, ouch, proud, so, spoken, stung, won't]
After merge()
lst1: []
lst2: []
answer: [a, another, baby, be, bringing, bumblebee, could, cutest,
face, got, home, i'm, it, little, me, mommy, my, not, of, one, ouch,
place, proud, so, spoken, stung, take, the, there's, won't, you've,
your]
BUILD SUCCESSFUL (total time: 0 seconds)
```