CSCM12 – Software concepts and efficiency March 3rd 2023
Tonicha Crook & Cécilia Pradic

Lab 4: doubly linked lists implementation

The goal of this lab is mostly to get you to play with a variant of linked lists, so as to familiarize yourself with programming with recursive structures in Java. To do so, we provide you with a template file `ListExercises.java` and a number of placeholder functions for you to fill in there (and some dummy `return` statement that you should get rid of ultimately - they are there so that the code compiles even if it is unfinished).

I suggest that you test your code whenever you have finished writing a function so that you spend a reasonable amount of time debugging. Also this time around I would suggest you all try to write your own copy of the code, even if you help each other otherwise - and archive it.

The questions below correspond to things to sequentially fill out in the template, which contains further instructions in comments regarding their specification. Question 1-3 are about the class `DLList<T>`, and question 4 is a more advanced one for people who have time to spare.

Happy hacking!

1. Write implementations for the methods

   ```
   public void push_back(T x);
   public void push_front(T x);
   public T pop_front();
   public void concatenate(DLList<T> xs);
   ```

   as specified in the template. You implementation should work in $\mathcal{O}(1)$.

2. Go back and uncomment the `length` attribute. Update your functions above so that it remains consistent. Then implement the methods

   ```
   public int size();
   public T get(int idx);
   public void insertAt(int idx, T x);
   ```

3. The goal of this question is to implement the merge sort algorithm over doubly-linked lists, which can be done with a linear space complexity (essentially there is never any need to duplicate the input if we are willing to throw it away).

   Implement the methods

   ```
   public DLList<T> splitInHalf();
   public void mergeIn(Comparator<T> cmp, DLList<T> other)
   public void mergeSort(Comparator<T> cmp)
   ```

4. Read up on `ListIterator<T>` at `https://docs.oracle.com/en/java/javase/18/docs/api/java.base/java/util/ListIterator.html`, and complete the implementation of a class satisfying that specification at the bottom of the file. That class should have a constructor that takes a `DLList<T>` and has all operations running in constant time.