HW-3

1.

Write each of the following *pure* functions (a function is pure if it does not update any of the arguments or global variables). No functions in the `Collections` class can be used.

1.1 The function `take(n, lst)` returns a list that takes `n` elements from `lst`.

```
public static <E> LinkedList<E> take(int n, LinkedList<E> lst)
```

1.2 The function `drop(n, lst)` returns a list that keeps all the elements of `lst` except for the first `n` elements. For example, `drop(5,[1,2,3,4,5,6,7,8,9,10])` returns `[6,7,8,9,10]`.

```
public static <E> LinkedList<E> drop(int n, LinkedList<E> lst)
```

1.3 The function `reverse(lst)` returns a copy of the list `lst` with the elements reversed.

```
public static <E> LinkedList<E> reverse(LinkedList<E> lst)
```

1.4 The function `sortedDown(lst)` checks if `lst` is sorted in *non-increasing* order. For example, for `lst = [3,3,2,1]`, it returns true.

```
public static <E extends Comparable<E>> boolean sortedDown(LinkedList<E> lst)
```

2.

Write a program that reads words from a text file and displays all the words (duplicates allowed) in ascending alphabetical order. The text file is passed as a command-line argument.

3.

A Java program contains various pairs of grouping symbols, such as:

- Parentheses: (and)
- Braces: and
- Brackets: [and]

Note that the grouping symbols cannot overlap. For example, `(a{b)}` is illegal. Write a program to check whether a Java source-code file has correct pairs of grouping symbols. Pass the source-code file name as a command-line argument.

4.

The *heap-sort* algorithm sorts a collection using the heap data structure. Since the heap data structure is used in the implementation of `PriorityQueue`, you can implement the heap-sort algorithm using `PriorityQueue`. Do the implementation:

```
public static <E extends Comparable<E>> LinkedList<E> heapSort(LinkedList<E> lst)
```

Notice that this is a pure function.

5. (project)

Write a method that converts an infix expression into a postfix expression using the following header:

```
public static String infixToPostfix(String expression)
```

For example, the method should convert the infix expression `(1 + 2) * 3` to `1 2 + 3 *` and `2 * (1 + 3)` to`2 1 3 + *`.