Objectives. Implement simple client programs that make use of the following basic data structures:

1. Linked lists.
2. Stacks.
3. Queues.

Problem 1. (*Min Max*) Implement the static methods `min()` and `max()` in `MinMax.java` that take a reference `first` to the first node in a linked list of integer-valued items as argument and returns the minimum and the maximum values respectively.

```
$ java MinMax
true
```

Problem 2. (*Matching Parentheses*) Implement the static method `match()` in `Parentheses.java` that takes a string `s` as argument and uses a stack to determine whether its parentheses are properly balanced, and returns `true` if they are and `false` otherwise. You may assume that `s` only consists of parentheses (curly, square, and round).

```
$ java Parentheses
[()]{}{[()()]()}
<ctrl-d>
true
$ java Parentheses
[(])
<ctrl-d>
false
```

Problem 3. (*Kth String from the End*) Write a `Queue` client `KthString.java` that takes a command-line argument k and prints the kth string from the end found on standard input, assuming that standard input has k or more strings.

```
$ java KthString 9
it was the best of times it was the worst of times
<ctrl-d>
best
```

Problem 4. (*Text Editor Buffer*) Develop a data type `Buffer` for a buffer in a text editor that implements the following API:

method	description
`Buffer()`	create an empty buffer
`void insert(char c)`	insert c at the cursor position
`char delete()`	delete and return the character at the cursor
`void left(int k)`	move the cursor k positions to the left
`void right(int k)`	move the cursor k positions to the right
`int size()`	number of characters in the buffer
`String toString()`	string representation of the buffer with a `'|'` character (not part of the buffer) at the cursor position

Hint: Use two stacks `left` and `right` to store the characters to the left and right of the cursor, with the characters on top of the stacks being the ones immediately to its left and right.

```
$ java Buffer
|There is grandeur in this view of life, with its several powers,
having been originally breathed by the Creator into a few forms or
into one; and that, whilst this planet has gone cycling on according
to the fixed law of gravity, from so simple a beginning endless forms
most beautiful and most wonderful have been, and are being, evolved.
-- Charles Darwin, The Origin of Species
```

Problem 5. (*Josephus Problem*) In the Josephus problem from antiquity, N people are in dire straits and agree to the following strategy to reduce the population. They arrange themselves in a circle (at positions numbered from 0 to $N-1$) and proceed around the circle, eliminating every Mth person until only one person is left. Legend has it that Josephus figured out where to sit to avoid being eliminated. Write a `Queue` client `Josephus.java` that takes N and M from the command line and prints out the order in which people are eliminated (and thus would show Josephus where to sit in the circle).

```
$ java Josephus 7 2
1 3 5 0 4 2 6
$ java Josephus 20 3
2 5 8 11 14 17 0 4 9 13 18 3 10 16 6 15 7 1 12 19
```

Files to Submit

1. `MinMax.java`
2. `Parentheses.java`
3. `Buffer.java`
4. `KthString.java`
5. `Josephus.java`

Before you submit:

- Make sure your programs meet the input and output specifications by running the following command on the terminal:

  ```
  $ python3 run_tests.py -v [<problems>]
  ```

  where the optional argument `<problems>` lists the problems (`Problem1`, `Problem2`, etc.) you want to test, separated by spaces; all the problems are tested if no argument is given.

- Make sure your programs meet the style requirements by running the following command on the terminal:

  ```
  $ check_style <program>
  ```

  where `<program>` is the `.java` file whose style you want to check.