Problem 1. (*Iterable Binary Strings*) Implement an immutable, iterable data type called `BinaryStrings` to systematically iterate over binary strings of length n. The data type must support the following API:

`BinaryStrings`

`BinaryStrings(int n)`	constructs an iterable `BinaryStrings` object given the length of binary strings needed
`Iterator<String> iterator()`	returns an iterator to iterate over binary strings of length n

`>_ ~/workspace/exercise2`

```
$ java BinaryStrings 3
000
001
010
011
100
101
110
111
```

Problem 2. (*Iterable Primes*) Implement an immutable, iterable data type called `Primes` to systematically iterate over the first n primes. The data type must support the following API:

`Primes`

`Primes(int n)`	constructs a `Primes` object given the number of primes needed
`Iterator<Integer> iterator()`	returns an iterator to iterate over the first n primes

`>_ ~/workspace/exercise2`

```
$ java Primes 10
2
3
5
7
11
13
17
19
23
29
```

Problem 3. (*Min Max*) Implement a library called `MinMax` with static methods `min()` and `max()` that accept a reference `first` to the first node in a linked list of integer-valued items and return the minimum and the maximum values respectively.

`>_ ~/workspace/exercise2`

```
$ java MinMax
min(first) == StdStats.min(items)? true
max(first) == StdStats.max(items)? true
```

Problem 4. (*Text Editor Buffer*) Implement a data type called `Buffer` to represent a buffer in a text editor. The data type must support the following API:

Buffer

`Buffer()`	creates an empty buffer
`void insert(char c)`	inserts `c` at the cursor position
`char delete()`	deletes and returns the character immediately ahead of the cursor
`void left(int k)`	moves the cursor `k` positions to the left
`void right(int k)`	moves the cursor `k` positions to the right
`int size()`	returns the number of characters in this buffer
`String toString()`	returns a string representation of this buffer with the `"|"` character (not part of the buffer) at the cursor position

```
>_ ~/workspace/exercise2
$ 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
```

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.

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 1 to n) 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. Implement a program `Josephus.java` that accepts n (int) and m (int) as command-line arguments, and writes to standard output the order in which people are eliminated (and thus would show Josephus where to sit in the circle).

```
>_ ~/workspace/exercise2
$ java Josephus 7 2
2
4
6
1
5
3
7
```

Files to Submit

1. `BinaryStrings.java`
2. `Primes.java`
3. `MinMax.java`
4. `Buffer.java`
5. `Josephus.java`

Before you submit your files, make sure:

- You do not use concepts outside of what has been taught in class.
- Your code is adequately commented, follows good programming principles, and meets any specific requirements such as corner cases and running times.