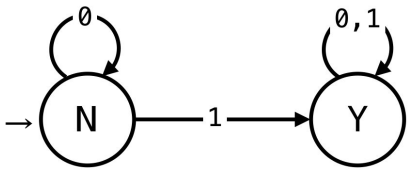
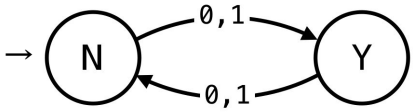
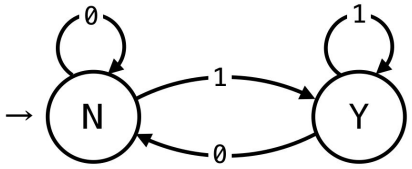
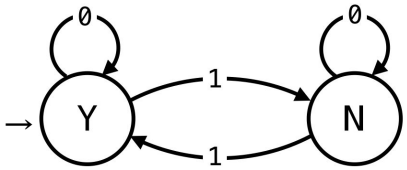
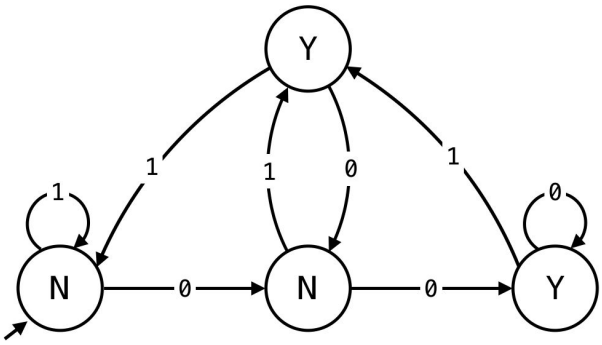
EXERCISE 1: Deterministic Finite Automata (DFA) Review

(a) Complete the following table to describe the set of binary strings accepted by each of the given DFAs.

The set of binary strings containing at least one `"1"`	
	
	
	
[optional]	*[optional]* 

(b) Provide a DFA that accepts only the set of all binary strings made of repetitions of `"01"`. *[optional]*
Examples: `"01"`, `"0101"`, `"01010101"`, `etc.`

EXERCISE 2: Binary Streams

public class BinaryStdIn	
boolean readBoolean()	*read 1 bit of data and return as a* `boolean` *value*
char readChar()	*read 8 bits of data and return as a* `char` *value*
char readChar(int r)	*read* `r` *(between 1 and 16) bits of data and return as a* `char` *value*
[*similar methods for* `byte` *(8 bits);* `short` *(16 bits);* `int` *(32 bits);* `long` *and* `double` *(64 bits)*]	
boolean isEmpty()	*is the bitstream empty?*
void close()	*close the bitstream*

public class BinaryStdOut	
void write(boolean b)	*write the specified bit*
void write(char c)	*write the specified 8-bit* `char`
void write(char c, int r)	*write the* `r` *(between 1 and 16) least significant bits of the specified* `char`
[*similar methods for* `byte` *(8 bits);* `short` *(16 bits);* `int` *(32 bits);* `long` *and* `double` *(64 bits)*]	
void close()	*close the bitstream*

Download `BinaryStreams.zip` from the precepts page and use the above API to perform the following:

(a) Extract from a **GIF** image the bits describing the of *number of bits per pixel* and print them to the screen as three binary digits. Test your code with the provided `test.gif` GIF image.

(b) Extract from a **GIF** image the byte that differentiates between different GIF versions (7 or 9). Print out this byte as a 32-bit integer, not as an ASCII character.

Use `HexDump.java` to see your output as follows:

```
% java-algs4 BinaryStreams < test.gif | java-algs4 HexDump 16
```

The following is an explanation for part of the file header in the GIF file format[1]:

> **GIF Header**
>
> ```
>      Offset   Length   Contents
>         0      3 bytes  "GIF"
>      →  3      3 bytes  GIF version: "87a" or "89a"
>         6      2 bytes  Screen Width
>         8      2 bytes  Screen Height
>        10      1 bit    Is there a Global Color Table?
>                3 bits   Number of bits of color resolution - 1.
>                1 bit    -
>      →         3 bits   Number of bits per pixel - 1.
>        11      1 byte   Index of Background Color in color table.
>        ...
> ```

[1] https://en.wikipedia.org/wiki/GIF#File_format

EXERCISE 3: Largest Anagram Set

Given a set of N English words, design an algorithm for finding the largest ***anagram set*** that can be constructed from these words. An ***anagram set*** is a set of words such that all words are anagrams of one another.

Example: `{tars, arts, rats, star}`.

Under reasonable assumptions, your algorithm should have a running time in the order of NL^2 or better, where L is the length of the longest English word.

Your algorithm should print to the screen the contents of *a* largest anagram set. If there is more than one, you may report any of them. Precisely describe any data structures you use.

EXERCISE 4: Longest Anagram Ladder

Given a set of N English words, design an algorithm for finding the longest ***anagram ladder***. An ***anagram ladder*** is a sequence of words such that the $k+1^{th}$ word is an anagram of the k^{th} word plus any character in the English alphabet.

Example: `{to, lot, lost, toils, tonsil, lotions, colonist, locations, coalitions, dislocation, conditionals, consolidation, consolidations}.`

Under reasonable assumptions, your algorithm should have a running time in the order of NL^2 or better, where L is the length of the longest English word.

Your algorithm should print to the screen the contents of *a* longest anagram ladder. If there is more than one, you may print any of them. Precisely describe any data structures you use.