Generate commands

```
$ ls
contact-de.html  faq-de.html  index-de.html  news-de.html  products-de.html
contact-en.html  faq-en.html  index-en.html  news-en.html  products-en.html
contact-fr.html  faq-fr.html  index-fr.html  news-fr.html  products-fr.html
$ mkdir en de fr # Create one directory for each language
$ ls |
> sed -n 's/\([^-]*\)-\(..\)\.html/mv \1-\2.html \2\/\1.html/p'
mv contact-de.html de/contact.html
mv contact-en.html en/contact.html
mv contact-fr.html fr/contact.html
mv faq-de.html de/faq.html
mv faq-en.html en/faq.html
mv faq-fr.html fr/faq.html
mv index-de.html de/index.html
mv index-en.html en/index.html
mv index-fr.html fr/index.html
mv news-de.html de/news.html
mv news-en.html en/news.html
mv news-fr.html fr/news.html
mv products-de.html de/products.html
mv products-en.html en/products.html
mv products-fr.html fr/products.html
$ ls | sed -n 's/\([^-]*\)-\(..\)\.html/mv \1-\2.html \2\/\1.html/p' | sh
$ ls
de  en  fr
$ ls en
contact.html  faq.html  index.html  news.html  products.html
$
```

Hold space

```
$ cat chapter.tex
\section{Introduction}
% TODO John

\section{Methods}
% TODO Mary

\section{Results}
% TODO Teresa

\section{Conclusions}
% TODO Fred
In this chapter we showed what we outlined in the introduction.
$ sed -e '
> /\\section/ {
>   h # Copy pattern space to hold space
>   d # Delete
> }
> /^% TODO/ {
>   H # Append pattern space to hold space
>   g # Copy hold space to pattern space
>   s/\n/ / # Remove embedded newline
> }
> ' chapter.tex
\section{Introduction} % TODO John

\section{Methods} % TODO Mary

\section{Results} % TODO Teresa

\section{Conclusions} % TODO Fred
In this chapter we showed what we outlined in the introduction.
```

```
$
```

Conditional statements

```
$ cat ntuples.txt
a = 3; b = 5; c = 8;
k = 8; j = 9;
$ cat >to-ntuple.sed <<\EOF
> #!/usr/bin/sed -f
> s/^/()();/ # Add two empty n-tuples
> :loop # Label
> # Move assignments to tuples
> s/(\([^)]*\))(\([^)]*\));\(.\) = \(.\); */(\1\3, )(\2\4, );/
> t loop # If successful repeat
> s/, )/)/g # Remove trailing commas
> s/)(/) = (/ # Add assignment
> EOF
$ chmod +x to-ntuple.sed # Make script executable
$ ./to-ntuple.sed ntuples.txt # Run script on ntuples.txt
(a, b, c) = (3, 5, 8);
(k, j) = (8, 9);
$
```

Towers of Hanoi

```
$ cat hanoi.sed
# Towers of Hanoi in sed.
#
#       @(#)hanoi.sed   5.1 (Berkeley) 10/10/90
#
#
# Ex:
# Run "sed -f hanoi.sed", and enter:
#
#       :abcd: : :<CR><CR>
#
# note -- TWO carriage returns, a peculiarity of sed), this will output the
# sequence of states involved in moving 4 rings, the largest called "a" and
# the smallest called "d", from the first to the second of three towers, so
# that the rings on any tower at any time are in descending order of size.
# You can start with a different arrangement and a different number of rings,
# say :ce:b:ax: and it will give the shortest procedure for moving them all
# to the middle tower.  The rules are: the names of the rings must all be
# lower-case letters, they must be input within 3 fields (representing the
# towers) and delimited by 4 colons, such that the letters within each field
# are in alphabetical order (i.e. rings are in descending order of size).
#
# For the benefit of anyone who wants to figure out the script, an "internal"
# line of the form
#               b:0abx:1a2b3 :2   :3x2
# has the following meaning: the material after the three markers :1, :2,
# and :3 represents the three towers; in this case the current set-up is
# ":ab :   :x  :".  The numbers after a, b and x in these fields indicate
# that the next time it gets a chance, it will move a to tower 2, move b
# to tower 3, and move x to tower 2.  The string after :0 just keeps track
# of the alphabetical order of the names of the rings.  The b at the
# beginning means that it is now dealing with ring b (either about to move
# it, or re-evaluating where it should next be moved to).
#
# Although this version is "limited" to 26 rings because of the size of the
# alphabet, one could write a script using the same idea in which the rings
# were represented by arbitrary [strings][within][brackets], and in place of
# the built-in line of the script giving the order of the letters of the
# alphabet, it would accept from the user a line giving the ordering to be
# assumed, e.g. [ucbvax][decvax][hplabs][foo][bar].
#
```

```
#                       George Bergman
#                       Math, UC Berkeley 94720 USA

# cleaning, diagnostics
s/  *//g
/^$/d
/[^a-z:]/{a\
Illegal characters: use only a-z and ":".  Try again.
d
}
/^:[a-z]*:[a-z]*:[a-z]*:$/!{a\
Incorrect format: use\
\       : string1 : string2 : string3 :<CR><CR>\
Try again.
d
}
/\([a-z]\).*\1/{a\
Repeated letters not allowed.  Try again.
d
}
# initial formatting
h
s/[a-z]/ /g
G
s/^:\( *\):\( *\):\( *\):\n:\([a-z]*\):\([a-z]*\):\([a-z]*\):$/:1\4\2\3:2\5\1\3:3\6\1\2:0/
s/[a-z]/&2/g
s/^/abcdefghijklmnopqrstuvwxyz/
:a
s/^\(.\).*\1.*/&\1/
s/.//
/^[^:]/ba
s/\([^0]*\)\(:0.*\)/\2\1:/
s/^[^0]*0\(.\)/\1&/
:b
# outputting current state without markers
h
s/.*:1/:/
s/[123]//gp
g
:c
# establishing destinations
/^\(.\).*\1:1/td
/^\(.\).*:1[^:]*\11/s/^\(.\)\(.*\1\([a-z]\).*\)\3./\3\2\31/
/^\(.\).*:1[^:]*\12/s/^\(.\)\(.*\1\([a-z]\).*\)\3./\3\2\33/
/^\(.\).*:1[^:]*\13/s/^\(.\)\(.*\1\([a-z]\).*\)\3./\3\2\32/
/^\(.\).*:2[^:]*\11/s/^\(.\)\(.*\1\([a-z]\).*\)\3./\3\2\33/
/^\(.\).*:2[^:]*\12/s/^\(.\)\(.*\1\([a-z]\).*\)\3./\3\2\32/
/^\(.\).*:2[^:]*\13/s/^\(.\)\(.*\1\([a-z]\).*\)\3./\3\2\31/
/^\(.\).*:3[^:]*\11/s/^\(.\)\(.*\1\([a-z]\).*\)\3./\3\2\32/
/^\(.\).*:3[^:]*\12/s/^\(.\)\(.*\1\([a-z]\).*\)\3./\3\2\31/
/^\(.\).*:3[^:]*\13/s/^\(.\)\(.*\1\([a-z]\).*\)\3./\3\2\33/
bc
# iterate back to find smallest out-of-place ring
:d
s/^\(.\)\(:0[^:]*\([^:]\)\1.*:\([123]\)[^:]*\1\)\4/\3\2\4/
td
# move said ring (right, resp. left)
s/^\(.\)\(.*\)\1\([23]\)\(.*:\3[^ ]*\) /\1\2 \4\1\3/
s/^\(.\)\(.*:\([12]\)[^ ]*\) \(.*\)\1\3/\1\2\1\3\4 /
tb
s/.*/Done!  Try another, or end with ^D./p
d
$ sed -f hanoi.sed
 :abcd: : :
:abcd:    :    :
```

```
:abc :    :d   :
:ab  :c   :d   :
:ab  :cd  :    :
:a   :cd  :b   :
:ad  :c   :b   :
:ad  :    :bc  :
:a   :    :bcd :
:    :a   :bcd :
:    :ad  :bc  :
:c   :ad  :b   :
:cd  :a   :b   :
:cd  :ab  :    :
:c   :ab  :d   :
:    :abc :d   :
:    :abcd:    :
Done!  Try another, or end with ^D.
$
```