Summer 2013 Handout 9

Simple sed applications: pp. 108−114

In **`vi`** we could say

```
:1,$s/Acme/Bongdex/g
```
First make sure you're not in insert mode.

To make **`sed`** do the same thing, give it the argument

```
s/Acme/Bongdex/g
```

with no leading **`:1,$`**.

The following **`sed`** commands do not change the contents of the file **`prog.c`**. They merely output a copy of what the file would look like if you changed all the **`Acme`**'s to **`Bongdex`**'s.

```
1$ sed 's/Acme/Bongdex/g' prog.c
2$ sed 's/Acme/Bongdex/g' prog.c | lpr
```

The following **`sed`** command does not change the contents of **`prog.c`** either. It merely destroys the contents of **`prog.c`**. See Handout 2, p. 22, "How not to redirect I/O".

```
3$ sed 's/Acme/Bongdex/g' prog.c > prog.c
```

To change the contents of **`prog.c`**, you must run two separate commands:

```
4$ sed 's/Acme/Bongdex/g' prog.c > ~/temp
5$ mv ~/temp prog.c
```
if you have **`w`** *permission for* **`prog.c`**

Make sure that there are no blanks or tabs after the **`s`** command (otherwise, you'll get **`command garbled`**.) To input from, and output to, the same file, see the **`-o`** option of **`sort`** and the **`-i`** option of Perl and Ruby.

```
6$ man -M /usr/perl5/man perlrun
```
Perl command line options

```
#!/bin/ksh
#Change every occurrence of the word Acme to Bongdex in every file in
#the current directory.  (Put shellscript in *different* directory.)

for filename in *
do
    echo $filename                  #Give the user something to watch.

    sed 's/Acme/Bongdex/g' $filename > ~/temp
    mv ~/temp $filename
done

exit 0
```

```
#!/bin/ksh
#Change every occurrence of the word Acme to Bongdex
#and every 2012 to 2013 in every file in the current directory.

for filename in *
do
    echo $filename

    sed '
        s/Acme/Bongdex/g
        s/2012/2013/g
    ' $filename > ~/temp

    mv ~/temp $filename
done

exit 0
```

You don't need two separate **sed** commands:

```
sed 's/Acme/Bongdex/g' $filename | sed 's/2012/2013/g' > ~/temp
```

Two ways to give sed a multi-line argument

The following rules hold for **sed**, **awk**, Perl, etc.

(1) As always, surround the argument of **sed** by single quotes. Neither **sed** nor the shell require the indentation.

http://i5.nyu.edu/~mm64/INFO1-CE9545/src/meatball

```
#!/bin/ksh

sed '
    #In some versions of sed, the argument can contain
    #only one comment line and no empty lines.

    s/Old Smoky/spaghetti/g
    s/snow/cheese/g;     #comment alongside needs semicolon too
    s/true lover/poor meatball/g
    s/For courting too slow/When somebody sneezed/g
'

exit 0
```

(2) You can write the argument of **sed** in a separate file called a *sedscript,* conventionally with a filename ending in **.sed**. A sedscript is not a shellscript: do not write the **#!/bin/ksh**, and do not use **chmod** to make the **.sed** file executable. The single quotes are gone too.

http://i5.nyu.edu/~mm64/INFO1-CE9545/src/meatball.sed

```
#This file is named meatball.sed.  This line is a comment.

s/Old Smoky/spaghetti/g
s/snow/cheese/g;     #comment alongside
s/sweetheart/meatball/g
s/For courting too slow/When somebody sneezed/g
```

Then on the command line we can say

```
    2$ sed -f ˜/meatball.sed infile > outfile
```

and in a shellscript we can say

```
#!/bin/ksh

sed -f ˜/meatball.sed ˜/infile > ˜/outfile
exit 0
```

awk has the same **-f** option.

▼ Homework 9.1: sedscript to convert Roman numerals to Arabic numerals

Insert six **s** commands into the following sedscript so that it will also translate Roman numerals that contain subtraction, i.e., **iv**, **ix**, **xl**, **xc**, **cd**, and **cm**. Insert these new commands in the right place in the list. Add the final **g** only if necessary. In the shellscript, change only the **$S45/roman.sed**.

http://i5.nyu.edu/~mm64/INFO1-CE9545/src/roman

```
#!/bin/ksh
#This shellscript, named roman, reads lines of input consisting of
#one Roman numeral per line, and outputs them as Arabic numerals,
#one per line.

tr '[IVXLCDM]' '[ivxlcdm]' | sed -f $S45/roman.sed | bc
exit 0
```

http://i5.nyu.edu/~mm64/INFO1-CE9545/src/roman.sed

```
#This sedscript, named roman.sed, converts Roman numerals to Arabic
#numerals.  Each line of input contains one lowercase Roman numeral.
#The output can be fed directly to the calculator program bc.  For
#example, ccxxxi becomes 100+100+10+10+10+1.

s/i/+1/g
s/v/+5/

s/x/+10/g
s/l/+50/

s/c/+100/g
s/d/+500/

s/m/+1000/g

#bc will not accept a plus at the start of a line.
s/ˆ+//
```

Here's how a typical line of input would be transformed:

CCXXXI	*original line of input*
ccxxxi	*after the* **tr**
+100+100+10+10+10+1	*after the* **s/m/+1000/g** *in the* **sed**
100+100+10+10+10+1	*after the* **sed**
231	*after the* **bc**

Test the shellscript like this:

```
    1$ roman < $S45/mmxiii
```

Print **$S45/mmxiii** and your output side-by-side with **pr -i' '1 -l1 -m -t -w80**. For these options of **pr**, see Handout 4, pp. 1−2.

▲

sed example: English to Pig Latin

We temporarily add a blank to the start of each line so that every word, even the first on a line, is immediately preceded by a character that is not a letter. This lets us use the regular expression **[^a-z]** to match what is immediately in front of every word.

The blank is scaffolding: text that we temporarily insert to make other editing commands simpler, like the leading zeroes in the **vi** Calisthenics I in Handout 8, p. 9. See Kernighan and Pike, *The Practice of Programming,* pp. 69 and 71: “As a rule, try to handle irregularities and exceptions and special cases in data. Code is harder to get right . . .”

See the “word boundary” **\b** in

```
    1$ man -M /usr/perl5/man perlre
```
Perl regular expressions

http://i5.nyu.edu/~mm64/INFO1-CE9545/src/piglatin

```
#!/bin/ksh
#Translate English to lowercase Pig Latin: owercaselay igpay atinlay
#Sample use:   piglatin < $S45/karenina | more

tr '[A-Z]' '[a-z]' |
sed '
    s/^/ /

#If the word starts with a vowel, temporarily prefix a "w".
    s/\([^a-z]\)\([aeiou]\)/\1w\2/g

#Chop off first letter of the word, move it to end, and add "ay".
    s/\([a-z]\)\([a-z]*\)/\2\1ay/g

    s/^ //
'

exit 0
```

```
2$ head -2 $S45/karenina
     Happy families are all alike; every unhappy family is
unhappy in its own way.

3$ head -2 $S45/karenina | piglatin
     appyhay amiliesfay areway allway alikeway; everyway unhappyway amilyfay isway
unhappyway inway itsway ownway ayway.
```

Great Moments in Computer History: The First Fortran Compiler (1957)

Some mail addresses have no obvious operator precedence, and thus are difficult to parse correctly. . . . [**Sendmail** ruleset 3] adds the magic tokens **<** and **>** [to an email address] to help set precedence.

—Evi Nemeth et al., *Unix System Administration Handbook, 1st ed.*, pp. 321−322

See

`http://en.wikipedia.org/wiki/Operator-precedence_parser#Alternatives_to_Dijkstra.27s_Algorithm`

—On the Web at
`http://i5.nyu.edu/~mm64/INFO1-CE9545/src/fortran.sed`

```
 1 #sedscript to fully parenthesize Fortran expressions
 2 #using binary + - * / ** (no unary + or -)
 3 #
 4 #   ( becomes (((      (line 27)
 5 #   ) becomes )))      (line 27)
 6 #
 7 #   + becomes ))+((    (line 29)
 8 #   - becomes ))-((    (line 29)
 9 #
10 #   * becomes )*(      (line 36)
11 #   / becomes )/(      (line 36)
12 #
13 #Finally, add (( and )) to the start and end of the line (line 41).
14 #For example,
15 #
16 #   A+B      becomes ((A))+((B))
17 #   A+B*C    becomes ((A))+((B)*(C))
18 #   A+B*C**D becomes ((A))+((B)*(C**D))
19 #   A**(B+C) becomes ((A**(((B))+((C)))))
20 #
21 #The algorithm adds more parentheses than are strictly necessary.
22 #It works correctly even if the input already has some parentheses.
23
24 #Triple each of the user’s parenthesis to overpower the ones inserted below.
25 #See Handout 8, p. 6 for the ampersand.  Why must this substitute command come
26 #before all the others that insert parentheses?
27 s/[()]/&&&/g
28
29 s/[+-]/))&((/g
30
31 #Temporarily change ** to @ so that the substitute command in line 36 will not
32 #mistake ** for two multiplication symbols.  Use \* to search for an asterisk.
33 s/\*\*/@/g
34
35 #* has no special meaning within [], so we need no backslash to turn it off.
36 s/[*/]/)&(/g
37
38 #Now that we’re safely past line 36, change exponentiation back to **.
39 s/@/**/g
40
41 s/.*/((&))/
```

sed example: reverse the order of the two operands on each line of assembly language

```
compiler | assembler
compiler | sed ’s/ \(.*\),\(.*\)$/ \2,\1/’ | assembler
```

```
*before*              *after*
mov r0,r1             mov r1,r0
add r2,_max           add _max,r2
cmp a,$1000           cmp $1000,a
```

sed example: rename all the files in the current directory

http://i5.nyu.edu/~mm64/INFO1-CE9545/src/rename

```
#!/bin/ksh
#Rename all the files in the current directory.  Do not change the
#contents of the files.  Compare Handout 5, pp. 20-21.

status=0   #innocent until proven guilty

for filename in *
do
    newname=`echo $filename | tr '[A-Z]' '[a-z]' | sed '
        s/[ ]/-/g
        s/^-\(.\)/\1/
        s/\.htm$/.html/
    '`

    if [[ -f $newname ]]
    then
        echo $0: found $filename and $newname 1>&2
        status=1
    else
        echo $filename
        mv $filename $newname
    fi
done

exit $status
```

sed example: normalize social security numbers

The following file contains haphazard social security numbers.

```
1$ more $S45/ss.txt
123-45-6789
123 45 6789
   123456789
SS Num 123-45-6789
123-45-678
000-00-0oOl
```

You can input the file into the following shellscript to output the numbers in a standard format:

http://i5.nyu.edu/~mm64/INFO1-CE9545/src/ss.normal

```
#!/bin/ksh

sed '
#Remove all characters except the digits.
        s/[^0-9]//g

#If the line does not consist of exactly 9 digits, change it
#to 9 question marks.  See p. 110 and Handout 8, p. 6 for !.
        /^.........$/!s/.*/?????????/

#Insert the two dashes.
        s/^\(...\)\(..\)/\1-\2-/
'

exit 0
```

```
2$ cd $S45
3$ pwd

4$ ss.normal < ss.txt
123-45-6789
123-45-6789
123-45-6789
123-45-6789
???-??-????
???-??-????
```

▼ Homework 9.2: normalize phone numbers

The file **`$S45/phone.txt`** contains haphazard phone numbers. Write a shellscript named **`phone.normal`** that will input the file and output the numbers in a standard format. For the arguments of **`pr`**, see Handout 4, pp. 1−2.

```
1$ phone.normal < $S45/phone.txt > phone.out
2$ pr -i' '1 -l1 -m -t $S45/phone.txt phone.out
(201) 200-1800                    (201) 200-1800
212-639-5555                      (212) 639-5555
2013433434                        (201) 343-3434
1-800-343-3434                    (800) 343-3434
555-1212                          (???) 555-1212
012-345-6789                      (???) 345-6789
122-345-6789                      (???) 345-6789
113-045-6789                      (113) ???-????
123-045-6789                      (???) ???-????
```

The argument of your **`sed`** should consist of exactly seven **`s`** commands:

(1) Remove all the non-digits.

(2) If the line consists of exactly seven characters, add three leading question marks to hold the place of the missing area code.

(3) If the line consists of a 1 followed by exactly ten characters, remove the leading 1.

(4) If the line now does not consist of exactly ten characters, change it to ten question marks.

(5) If the first digit of the area code is a 0, or the middle digit of the area code is neither a 1 nor a 0, change the area code to three question marks. Use one **`s`** command with the **`!`** on p. 110 and in Handout 8, p. 6, column 2.

(6) If the first digit of the phone number is a 0 or a 1, change the phone number to seven question marks.

(7) Add the parentheses, blank, and dash.

▲

sed example: insert the vt100 escape codes for color into a directory listing.

For the `.\{53\}`, see Handout 6, p. 9, line 11.

http://i5.nyu.edu/~mm64/INFO1-CE9545/src/ls

```
#!/bin/ksh
#List directories in blue using the vt100 terminal excape sequences.

esc=‘echo ’\033’‘   #the escape character
b="$esc[34m"        #begin blue
e="$esc[0m"         #end blue

#\1 is the first 54 characters on each line that starts with d.
#\2 is the name of the directory.
#Surround the name of the directory with "begin blue" and "end blue".

/bin/ls -l $* |
sed ’s/^\(d.\{53\}\)\(.*\)/\1’$b’\2’$e’/’

exit 0
```

sed example: edit a command line argument of a shellscript

Name this shellscript **chmod** and put it in your **bin** directory to create your own version of the Unix **chmod** command.

—On the Web at
http://i5.nyu.edu/~mm64/INFO1-CE9545/src/chmod

```
#!/bin/ksh
#New version of chmod envisioned on p. 56.  The user can write the 9
#permission bits as the 9 characters familiar from the output of ls -l,
#rather than as 3 octal digits.  Sample use: instead of
#   chmod 644 filename1 filename2 filename3 ...
#just say
#   chmod rw-r--r-- filename1 filename2 filename3 ...
#or even
#   chmod rw-r--r filename1 filename2 filename3 ...

if [[ $# -lt 2 ]]
then
    echo $0: must have at least two arguments. 1>&2
    exit 1
fi

octal=$1
shift              #Handout 7, p. 6.  Now $* will be only the filenames.

/bin/chmod ‘echo $octal | sed ’

#If the user typed less than 9 characters, right-pad the string
#with dashes until it is 9 characters long.
```

```
    s/$/---------/
    s/^\(.........\).*$/\1/

#If s/---/0/g were first, would change rwxr---wx to 7r0wx and get stuck.
    s/--x/1/g
    s/-w-/2/g
    s/-wx/3/g
    s/r--/4/g
    s/r-x/5/g
    s/rw-/6/g
    s/rwx/7/g
    s/---/0/g
'` $*              #Handout 4, pp. 21, 23 for $*

if [[ -t 1 ]]      #If the standard output is going to screen of terminal,
then
    ls -l $*       #give the user some feedback.
fi

exit 0
```

▼ **Homework 9.3: user-friendlier version of Homework 6.5 (Handout 6, pp. 11−12)**

Write a shellscript named **`spelledby2`** that takes one phone number as its command line argument and outputs all the words in **`/usr/dict/words`** and **`/usr/dict/websters`** that are spelled by that number. For example,

```
1$ spelledby2 7373783
reserve
```

Make sure that there is exactly one command line argument

`if [[ $# -ne` *etc.*

and that it contains no characters other than **`23456789`**:

`if [[ $1 == *[!2-9]*` *etc.: wildcard in Handout 7, pp. 2, 6;* **`ksh`***(1) p. 18*

Then **`echo`** the first argument into **`sed`**, and give **`sed`** a list of **`s`** commands such as

```
s/2/[abc]/g
```

to change each digit to a wildcard. There should also be an **`s`** command to remove non-digits; where in the list should it go? Finally, remove duplicates from the output of **`grep`**.

The **`‘`**back-quoted**`‘`** pipeline must be inside **`"`**double quotes**`"`** to prevent the shell from applying the wildcards output by **`sed`** to the names of the files in the current directory. If the **`‘`**back-quoted**`‘`** pipeline was in **`’`**single quotes**`’`**, the **`‘`**back quotes**`‘`** would not work at all.

```
#!/bin/ksh
set -x                  #Handout 5, pp. 8-9
cd /usr/dict

#The 1st argument of this grep is the word "words",
#because that is the only filename in /usr/dict that
#matches the shell wildcards [wxy][mon][prs][def][prs].
grep ‘echo 96737 | sed ’
    s/3/[def]/g
    s/6/[mno]/g
    s/7/[prs]/g
    s/9/[wxy]/g
’‘ words

#The 1st argument of this grep is a regular expresion w/ 5 wildcards.
grep "‘echo 96737 | sed ’
    s/3/[def]/g
    s/6/[mno]/g
    s/7/[prs]/g
    s/9/[wxy]/g
’‘" words

exit 0
```

```
    + cd /usr/dict
    + echo 96737
    + sed $’0/3/[def]/g0/6/[mno]/g0/7/[prs]/g0/9/[wxy]/g0
    + grep words words
    + echo 96737
    + sed $’0/3/[def]/g0/6/[mno]/g0/7/[prs]/g0/9/[wxy]/g0
    + grep ’[wxy][mno][prs][def][prs]’ words
    swordplay
    + exit 0
```

The ˆ and **$** are within **’**single quotes**’** to deprive them of any special meaning they have to the shell.

```
cd /usr/dict
grep -hi ’ˆ’"‘echo $1 | sed ’list of substitute commands’‘"’$’ words websters
```

If your phone number spells no words, run **spelledby2** separately on the first three digits and the last four digits, etc. For example,

```
2$ spelledby2 7373                    3$ spelledby2 783
Sere                                  pud
sere                                  rud
serf                                  rue
                                      sud
                                      sue
```

No change to **spelledby2** should be necessary to run it on a number with more or less than seven digits. ▲

See the grades: pp. 54−55

If a program’s **s** bit (set-uid) and **x** bit are on, you can execute it even though it contains commands that normally only the owner of the program is allowed to execute. (In some versions of Unix, this works only with C programs, not shellscripts.) For the set-uid bit, see Handout 6, p. 14, Homework 6.7. For a list of all twelve permission bits, see **chmod**(1) for the **chmod** program, **chmod**(2) for the **chmod** system call. Or see the **S_I** macros in the file **/usr/include/sys/stat.h**.

```
1$ ~mm64/bin/allgrades 45 | more
```

```
#!/bin/ksh
#This set-uid shellscript is ~mm64/bin/allgrades.
#List the students in the class specified by the command line argument,
#one per line but without their names, giving the homework numbers for
#which they have gotten credit.

IFS=' '   #Internal field separators, pp. 157-8

if [[ $# -ne 1 ]]
then
    /bin/echo $0: requires one command line arg 1>&2
    exit 1
fi

if [[ $1 -ne 45 && $1 -ne 47 &&
    $1 -ne 64 && $1 -ne 65 && $1 -ne 66 && $1 -ne 70 ]]
then
    /bin/echo $0: command line arg must be one of \
        45 47 64 65 66 70 1>&2
    exit 2
fi

/bin/sed 's/^[^:]*://' ~mm64/$1/grades/homework
exit 0
```

```
2$ cd ~mm64/bin
3$ chmod 4555 allgrades
4$ ls -l allgrades
-r-sr-xr-x   1 mm64     users        647 Nov 22  2000 allgrades
```

at (Handout 7, p. 7) copies your program into the following directory. But the directory is owned by the superuser, and only the superuser has permission to copy a file into it:

```
5$ cd /var/spool/cron/atjobs
6$ pwd
7$ ls -ld
drwxr-xr-x   2 root     sys            2 May 28 15:22 .

8$ which at
/bin/at

9$ cd /bin
10$ ls -l at
-rwsr-xr-x   1 root     sys        54676 Jul  5  2012 at
```

Before turning on the **s** bit, do all of the following:

(1) Turn off the last two **w** bits. Is the parenthesized sentence on p. 55 true on i5.nyu.edu and your Unix system at work?

(2) Write out the full pathname of every program that you call.

(3) Use no environment variables: **$HOME**, **$PATH**, etc. See **environ**(5). Validate all command line arguments before you use them.

sed example: sort playing cards in order of increasing rank

Suppose you have a file of playing cards, one per line. The first column is the rank and the second column is the suit:

```
A           S
2           C
Q           H
J           D
```

http://i5.nyu.edu/~mm64/INFO1-CE9545/src/poker

```
#!/bin/ksh
#Sort playing cards, one per line, in order of increasing rank.
#Ignore the suits.
#Add "14 " to start of every line that begins with "A" or "a".
#See bystanders in Handout 8, p. 6, column 2.
#For & in an s/// command, see Hand 8, pp. 6-7; textbook pp. 323-324.

sed '
    s/^[2-9]/& &/
    s/^10/& &/
    s/^[Jj]/11 &/
    s/^[Qq]/12 &/
    s/^[Kk]/13 &/
    s/^[Aa]/14 &/
' |
sort -n |
sed 's/^[^ ]* //'   #Remove everything up to & including 1st blank.

exit 0
```

```
#!/bin/ksh
#Sort playing cards, one per line, in order of increasing rank.
#Ignore the suits.
#Add "14 " to start of every line that begins with "A" or "a".

awk '
    /^([2-9]|10)/ {print $1, $0}   #Handout 7, p. 11 for |
    /^[Jj]/       {print 11, $0}
    /^[Qq]/       {print 12, $0}
    /^[Kk]/       {print 13, $0}
    /^[Aa]/       {print 14, $0}
' |
sort -n |
sed 's/^[^ ]* //'  #Remove everything up to & including the 1st blank.

exit 0
```

After the first **sed**, our data is:

```
14 A        S
2 2         C
12 Q        H
11 J        D
```

After the **`sort -n`**, our data is:

```
2 2         C
11 J        D
12 Q        H
14 A        S
```

After the second **`sed`**, our data is:

```
2           C
J           D
Q           H
A           S
```

▼ Homework 9.4: sort whatever you want

Write a shellscript named **`customsort`** that sorts its lines of input into an order other than alphabetical or numerical. For example, if your input consists of one chemical element per line, sort them in order of increasing atomic number. Just do the first ten elements: **`H`**, **`He`**, **`Li`**, **`Be`**, **`B`**, **`C`**, **`N`**, **`O`**, **`F`**, and **`Ne`**.

before	*after*
`H`	`H`
`O`	`He`
`He`	`Li`
`Li`	`B`
`B`	`O`
`O`	`O`

▲

A safe way to display arbitrary text on the web

`<`, **`>`**, and **`&`** are special characters in HTML. See Handout 3, p. 9, lines 11−15. If an environment variable contains **`<`**, **`>`**, or **`&`**, the browser would become confused when displaying the output of **`env`**. Here's a safe way to do it.

We must change every **`&`** to an **`&amp;`** *before* we change every **`<`** to a **`&lt;`** and every **`>`** to a **`&gt;`**. We must surround each line with **`<LI>`** and **`</LI>`** *after* we change every **`<`** to a **`&lt;`** and every **`>`** to a **`&gt;`**.

An ampersand is a special character in the replacement section of a substitute command in **`vi`** and **`sed`**. See Handout 8, p. 7, first line 2. We therefore need a backslash in front of each one that represents a plain old ampersand.

`sort` assumes that the contents of each environment variable is a single line. Is this always true?

http://i5.nyu.edu/~mm64/INFO1-CE9545/src/environment

```
#!/bin/ksh

echo ’Content-type: text/html’
echo
echo ’<HTML>’
echo ’<HEAD>’
echo ’<TITLE>Environment Variables</TITLE>’
echo ’</HEAD>’
echo ’<BODY>’
echo ’<H1>Environment Variables</H1>’
echo Here are all the environment variables that the web server
echo passed to the gateway.
echo ’<OL>’    #ordered list: Handout 3, p. 10, lines 89-94

env |
grep ’=’ |
sort -df |    #dictionary order, fold: Handout 1, p. 3, line 46
sed ’
    s/&/\&amp;/g
    s/</\&lt;/g
    s/>/\&gt;/g
    s:.*:<LI>&</LI>:
’

echo ’</OL>’
echo ’</BODY>’
echo ’</HTML>’
```

```
1$ export INEQUALITY=’a<b’
```
Create an environment variable; Handout 4, p. 4.

```
2$ echo $INEQUALITY
a<b

3$ environment | grep INEQUALITY
<LI>INEQUALITY=a&lt;b</LI>
```

Insert HTML tags into the output of a gateway (cf. Handout 5, pp. 27−29)

```
#!/bin/ksh
#Gateway script to output the loginnames of the classmates logged in.

echo Content-type: text/html
echo
echo ’<HTML>’
echo ’<HEAD>’
echo ’<TITLE>Classmates logged in</TITLE>’
echo ’</HEAD>’
echo ’<BODY>’
echo ’<H1>Classmates logged in</H1>’
echo ’<OL>’

˜mm64/bin/roster 45 | sort > /tmp/inclass$$
who | awk ’{print $1}’ | sort | uniq > /tmp/loggedin$$

comm -12 /tmp/inclass$$ /tmp/loggedin$$ |
sed ’s:.*:<LI>&</LI>:’

rm /tmp/inclass$$ /tmp/loggedin$$

echo ’</OL>’
echo ’</BODY>’
echo ’</HTML>’
exit 0
```

```
Content-type: text/html

<HTML>
<HEAD>
<TITLE>Classmates logged in</TITLE>
</HEAD>
<BODY>
<H1>Classmates logged in</H1>
<OL>
<LI>abc1234</LI>
<LI>def5678</LI>
<LI>ghi0912</LI>
</OL>
</BODY>
</HTML>
```

```
Classmates logged in

1. abc1234
2. def5678
3. ghi9012
```

Insert links into the output of a gateway

See Handout 8, pp. 6−7 (or pp. 323−324 in the textbook) for **&** in the replacement part of an **s** command.

```
#!/bin/ksh
#Gateway script to output the loginnames of the classmates logged in.

echo Content-type: text/html
echo
echo '<HTML>'
echo '<HEAD>'
echo '<TITLE>Classmates logged in</TITLE>'
echo '</HEAD>'
echo '<BODY>'
echo '<H1>Classmates logged in</H1>'
echo "Click on a loginname to go to that person's home page."
echo '<OL>'

~mm64/bin/roster 45 | sort > /tmp/inclass$$
who | awk '{print $1}' | sort | uniq > /tmp/loggedin$$

comm -12 /tmp/inclass$$ /tmp/loggedin$$ |
sed 's:.*:<LI><A HREF = "/~&/">&</A></LI>:'

rm /tmp/inclass$$ /tmp/loggedin$$

echo '</OL>'
echo '</BODY>'
echo '</HTML>'
exit 0
```

```
Content-type: text/html

<HTML>
<HEAD>
<TITLE>Classmates logged in</TITLE>
</HEAD>
<BODY>
<H1>Classmates logged in</H1>
Click on a loginname to go to that person's home page.
<OL>
<LI><A HREF = "/~abc1234/">abc1234</A></LI>
<LI><A HREF = "/~def5678/">def5678</A></LI>
<LI><A HREF = "/~ghi9012/">ghi9012</A></LI>
</OL>
</BODY>
</HTML>
```

```
Classmates logged in

Click on a loginname to go to that person’s home page.
1. abc1234
2. def5678
3. ghi9012
```

Another example is the gateway `~mm64/public_html/cgi-bin/cvs_log`.

▼ Homework 9.5: write a gateway with sed

Write a gateway that doesn’t always produce exactly the same output each time you run it. It can’t be the gateway shown above that outputs the loginnames of the people in the class who are logged in now. Put a **`sed`** in the gateway as in the above examples to insert HTML tags into the output of the gateway. Examples would be the list tags **`<LI>`** and **`</LI>`**, or the table and data tags **`<TR>`** and **`</TR>`**, **`<TD>`** and **`</TD>`**.

Hand in a printout of your **`index.html`** file, your gateway, and the standard output of your gateway. Circle the link in your home page that leads to your gateway.

▲

□