Unix Solved Examples

1> to delete 1st character of each line.
```
$ sed 's/^.//' file1
```

2> to delete 1st 2 characters of each line.
```
$ sed 's/^..//' file1
```

3> Delete last 3 lines of a file using sed
```
$ sed '$d' file1 | sed '$d' | sed '$d'
```
4> Frequency of vowels of a file
```
$sed 's/[^aeiouAEIOU]//g' file1 | sed 's/[aeiouAEIOU]/& /' | sed 'y/ /\n/' | sort | uniq -c
```

5> to display only first charecter of file
```
$sed -n 's/^.* //p' f2
```

6> during dealing with charecter

```
if [ $i == $j ]
then                         // space will be there after == and before ==
echo1
else
echo2
fi
```

Answers

1. Count number of characters in first five lines of file x1.
-> `$head -5 x1|wc -c`

2. Display files of current directory whose 1st character is not digit.
-> `$ls[a-zA-Z]*.*`

3. Display last 20 lines of working directory.
-> `$ls|tail -20`

4. Remove a file forcibly which do not have write permissions.
-> ``$rm -f `find -prem 444` ``

5. Display only those files of current directory which is own by the current user.
-> ``$find -user `who am i` -print``

6. To run a utility x1at 5:00 pm.
-> `$at 5:00 x1`

7. To replace the word computer with computing of file x1.

-> `$sed 's/computer/computing/g' x1`

8. Display lines where last word is UNIX of file x1.

-> `$grep 'unix$x1'`

9. Display number of lines having odd number of word in file x1.

-> `$`

10. To return back user to his/her home directory without using home directory name as an argument to that command.

-> `$cd`

11. Display lines at file x1 that do not begin with (^).

-> `$grep ^[^^]x1`

12. Display only hidden files of working directory.

-> `$ls -a`

13. Display lines starting from 10th line to end of file x1.

-> `$sed -n '10,$p' x1`

14. To concatenate two files f1 and f2 including some text to be inserted from keyboard after content of f1.

-> `$cat>>f1;cat f1>>f3;cat f2>>f3`

15. To assign output of ls command to positional parameter.

-> `$ls>>x3`

16. To delete all vowels from the file.

-> `$sed 's/[AaEeIiOoUu]//g' x1`

17. To open last modified file for editing purpose.

-> ``$vi `ls -t|head -d` ``

18. To display calendar for the current year.

-> `$cat -y`

19. To display all files of parent directory without moving parent.

-> `$ls..`

20. To copy a file x**y sin a parent directory.

-> `$cp x**y..`

21. To display all file names having last character as a digit.

-> `$ls -1|grep '[0-9$]'si`

22. To remove a file named as –x (-character followed by x).

-> `$rm___x`

23. To send off line message to user on terminal tty2.
-> $write exam50 tty2

24. To list files of working directory. Do not use command.
-> $echo *

25. To display common lines in x1 and x2.
-> $comm -2 x1 x2

26. To display last 10 lines of file x1.
-> $tail -10 x1

27. To run script x1 in background so that its execution continues user logout from the system.
-> $nohup sh x1 &

28. To sort files with country as the primary key and state as secondary key when a file consists of name of city state and country in the order.
-> $sort -t\R f1 f2

29. To grant read permission to group only of file x1.
-> $chmod g=tr x1

30. To display number of process to related to logging user.
-> $ps -T

31. To divide a file x1 into three files each containing.
-> $split -20 x1

32. To count number of character in a file x1 do not use wc.
-> $

33. To display number of logging users.
-> $who|wc -l

34. To create null file named as x1 in working directory do not use touch command.
-> $cat>x1

35. To remove duplicate lines from a file.
-> $sort -u x1

36. To kill the most recent background process whom neither PID nor job is known.
-> $kill $!

37. To remove a file named as x? (x followed by ?).
-> $rm x?

38. To send online message to all login users.
-> $wall "Hi"

39. To deny execute permission to group of file x1.
```
->   $chmod g=trw x1
```

40. To create a link between file x1 and x1.link.
```
->   $ln -s x1 x1.lnk
```

41. To combine content of file x1 do not use cat command.
```
->   $more x1
```

42. To display character from 5th position to 10th position including both from file x1.
```
->   $sed -n '5,10p' x1
```

43. To display 2nd and 3rd character of variable x.
```
->   $echo $1 `expr $x:'.||(..\)'`
```

44. To list file names consists of only digits
```
->   $ls|grep '^[0-9]*[0-9]*[0-9]$'
```

45. To assign output as well as message of a command x into a file x1.
```
->   $cat x1>x2 2>err.txt
```

46. To remove file named as x* to xy.
```
->   $mv x* xy
```

47. To count number regular files.
```
->   $find type -f|wc -l
```

48. To display line beginning with alphabets of file x1.
```
->   $grep '^[A-Za-z]' x1
```

49. To delete 3 consecutive lines from the current line of a file open by vi editor.
```
->   press Esc then 3 number and two tymes press dd
```

50. To display only those lines having only capital alphabets.
```
->   $grep '^[A-Z]*[A-Z]*[A-Z]$' x1
```

51. To copy file x1 into x2 don't use cp command.
```
->   $cat x1>x2
```

52. To display number of lines in a file don't use wc command.
```
->   $grep -c f1
```

53. To send a message to terminal tty1 using echo command.
```
->   $echo "Hello"|write tty1
```

54. Search all lines in a file x1 which end with a semicolon.
```
->   $grep '.;$' x1
```

55. Convert all capital letter in a file x1 to small case letters.
```
->  $cat x1|tr '[A-Z]''[a-z]'
```

56. Write a command to count number of blank lines in a file x1.
```
->  $grep -c '^$' x1
```

57. Append three records to input of emp.lst and store the new output in empnew.lst.
```
->  $cat>>emp.lst;cat emp.lst>>empnew.lst
```

58. Change the modification of time of a file midnight at 01/01/2006.
```
->  $
```

59. Merge the contents of file a.txt, b.txt and c.txt sort them and display the sorted output on screen page by page.
```
->  $sort -m a.txt b.txt c.txt
```

60. List all files beginning with character ‘P’ on the screen twice in succession.
```
->  $ls -l|grep '^p'|sed P
```

Answers

1. To double space a file x1 (do not include blank line before 1st line and after last line).
```
->  $sed 's///g' x1
```

2. To remove all extra spaces and a clank line from the file x1.
```
->  $sed -e 's///g' x1 -e '/^[]*$' x1
```

3. To store regular file name is reg.txt and directory file name in dir.txt of current directory.
```
->  $find *-lyptr>reg.tty; find * typed>dir.txt
```

4. To remove all alphabets from the file x1.
```
->  $sed 's/[A-Za-z]//g' x1
```

5. To display the content of file x1.
```
->  $sed -n '/,$p' x1
```

6. To display number of lines of file x1.
```
->  $sed -n x1|tail -l
```

7. To display lines from 3rd to 5th including both from file f1.
```
->  $sed sed -n '3,5p' x1
```

8. Swaps the first and last character in each line of x1.
```
->  $sed
```

9. To display the first words of each line of file x1.
```
->  $sed
```

10. To display last line of file x1.

-> `$sed -n '$p' x1`

11. To display 3 lines starting from 5th line of file x1.

-> `$sed -n '5,7p' x1`

12. To display before string unix from file x1.

-> `$sed '^unix' x1`

13. To display all blank lines of file x1.

-> `$sed '/^[]*$' x1`

14. To display lines beginning either with alphabets or digit from x1.

-> `$sed -n '/^[A-Za-z0-9]/p' x1`

15. Differentiate between command sed ‘s/Good/fine/g’ x1 and sed ‘s/good/fine’ x1.

-> `$sed 's/Good/fine/' x1`

16. To display lines from 5 to 10 including both.

-> `$sed '5,10p' x1`

FILE SPACING:

```
# double space a file
sed G

# double space a file which already has blank lines in it. Output file
# should contain no more than one blank line between lines of text.
sed '/^$/d;G'

# triple space a file
sed 'G;G'

# undo double-spacing (assumes even-numbered lines are always blank)
sed 'n;d'

# insert a blank line above every line which matches "regex"
sed '/regex/{x;p;x;}'

# insert a blank line below every line which matches "regex"
sed '/regex/G'

# insert a blank line above and below every line which matches "regex"
sed '/regex/{x;p;x;G;}'
```

NUMBERING:

```
 # number each line of a file (simple left alignment). Using a tab (see
 # note on '\t' at end of file) instead of space will preserve margins.
 sed = filename | sed 'N;s/\n/\t/'

 # number each line of a file (number on left, right-aligned)
 sed = filename | sed 'N; s/^/     /; s/ *\(.\{6,\}\)\n/\1  /'

 # number each line of file, but only print numbers if line is not blank
 sed '/./=' filename | sed '/./N; s/\n/ /'

 # count lines (emulates "wc -l")
 sed -n '$='
```

TEXT CONVERSION AND SUBSTITUTION:

```
 # IN UNIX ENVIRONMENT: convert DOS newlines (CR/LF) to Unix format.
 sed 's/.$//'               # assumes that all lines end with CR/LF
 sed 's/^M$//'              # in bash/tcsh, press Ctrl-V then Ctrl-M
 sed 's/\x0D$//'            # works on ssed, gsed 3.02.80 or higher

 # IN UNIX ENVIRONMENT: convert Unix newlines (LF) to DOS format.
 sed "s/$/`echo -e \\\r`/"            # command line under ksh
 sed 's/$'"/`echo \\\r`/"             # command line under bash
 sed "s/$/`echo \\\r`/"               # command line under zsh
 sed 's/$/\r/'                        # gsed 3.02.80 or higher

 # IN DOS ENVIRONMENT: convert Unix newlines (LF) to DOS format.
 sed "s/$//"                          # method 1
 sed -n p                             # method 2

 # IN DOS ENVIRONMENT: convert DOS newlines (CR/LF) to Unix format.
 # Can only be done with UnxUtils sed, version 4.0.7 or higher. The
 # UnxUtils version can be identified by the custom "--text" switch
 # which appears when you use the "--help" switch. Otherwise, changing
 # DOS newlines to Unix newlines cannot be done with sed in a DOS
 # environment. Use "tr" instead.
 sed "s/\r//" infile >outfile         # UnxUtils sed v4.0.7 or higher
 tr -d \r <infile >outfile            # GNU tr version 1.22 or higher

 # delete leading whitespace (spaces, tabs) from front of each line
 # aligns all text flush left
 sed 's/^[ \t]*//'                    # see note on '\t' at end of file

 # delete trailing whitespace (spaces, tabs) from end of each line
 sed 's/[ \t]*$//'                    # see note on '\t' at end of file
```

```
# delete BOTH leading and trailing whitespace from each line
sed 's/^[ \t]*//;s/[ \t]*$//'

# insert 5 blank spaces at beginning of each line (make page offset)
sed 's/^/     /'

# align all text flush right on a 79-column width
sed -e :a -e 's/^.\{1,78\}$/ &/;ta' # set at 78 plus 1 space

# center all text in the middle of 79-column width. In method 1,
# spaces at the beginning of the line are significant, and trailing
# spaces are appended at the end of the line. In method 2, spaces at
# the beginning of the line are discarded in centering the line, and
# no trailing spaces appear at the end of lines.
sed  -e :a -e 's/^.\{1,77\}$/ & /;ta'                     # method 1
sed -e :a -e 's/^.\{1,77\}$/ &/;ta' -e 's/\( *\)\1/\1/' # method 2

# substitute (find and replace) "foo" with "bar" on each line
sed 's/foo/bar/'             # replaces only 1st instance in a line
sed 's/foo/bar/4'            # replaces only 4th instance in a line
sed 's/foo/bar/g'            # replaces ALL instances in a line
sed 's/\(.*\)foo\(.*foo\)/\1bar\2/' # replace the next-to-last case
sed 's/\(.*\)foo/\1bar/'            # replace only the last case

# substitute "foo" with "bar" ONLY for lines which contain "baz"
sed '/baz/s/foo/bar/g'

# substitute "foo" with "bar" EXCEPT for lines which contain "baz"
sed '/baz/!s/foo/bar/g'

# change "scarlet" or "ruby" or "puce" to "red"
sed 's/scarlet/red/g;s/ruby/red/g;s/puce/red/g' # most seds
gsed 's/scarlet\|ruby\|puce/red/g'               # GNU sed only

# reverse order of lines (emulates "tac")
# bug/feature in HHsed v1.5 causes blank lines to be deleted
sed '1!G;h;$!d'               # method 1
sed -n '1!G;h;$p'             # method 2

# reverse each character on the line (emulates "rev")
sed '/\n/!G;s/\(.\)\(.*\n\)/&\2\1/;//D;s/.//'

# join pairs of lines side-by-side (like "paste")
sed '$!N;s/\n/ /'

# if a line ends with a backslash, append the next line to it
sed -e :a -e '/\\$/N; s/\\\n//; ta'
```

```
# if a line begins with an equal sign, append it to the previous line
# and replace the "=" with a single space
sed -e :a -e '$!N;s/\n=/ /;ta' -e 'P;D'

# add commas to numeric strings, changing "1234567" to "1,234,567"
gsed ':a;s/\B[0-9]\{3\}\>/,&/;ta'                     # GNU sed
sed -e :a -e 's/\(.*[0-9]\)\([0-9]\{3\}\)/\1,\2/;ta' # other seds

# add commas to numbers with decimal points and minus signs (GNU sed)
gsed -r ':a;s/(^|[^0-9.])([0-9]+)([0-9]{3})/\1\2,\3/g;ta'

# add a blank line every 5 lines (after lines 5, 10, 15, 20, etc.)
gsed '0~5G'                  # GNU sed only
sed 'n;n;n;n;G;'             # other seds
```

SELECTIVE PRINTING OF CERTAIN LINES:

```
# print first 10 lines of file (emulates behavior of "head")
sed 10q

# print first line of file (emulates "head -1")
sed q

# print the last 10 lines of a file (emulates "tail")
sed -e :a -e '$q;N;11,$D;ba'

# print the last 2 lines of a file (emulates "tail -2")
sed '$!N;$!D'

# print the last line of a file (emulates "tail -1")
sed '$!d'                    # method 1
sed -n '$p'                  # method 2

# print the next-to-the-last line of a file
sed -e '$!{h;d;}' -e x              # for 1-line files, print blank line
sed -e '1{$q;}' -e '$!{h;d;}' -e x  # for 1-line files, print the line
sed -e '1{$d;}' -e '$!{h;d;}' -e x  # for 1-line files, print nothing

# print only lines which match regular expression (emulates "grep")
sed -n '/regexp/p'           # method 1
sed '/regexp/!d'             # method 2

# print only lines which do NOT match regexp (emulates "grep -v")
sed -n '/regexp/!p'          # method 1, corresponds to above
sed '/regexp/d'              # method 2, simpler syntax

# print the line immediately before a regexp, but not the line
# containing the regexp
```

```
sed -n '/regexp/{g;1!p;};h'

# print the line immediately after a regexp, but not the line
# containing the regexp
sed -n '/regexp/{n;p;}'

# print 1 line of context before and after regexp, with line number
# indicating where the regexp occurred (similar to "grep -A1 -B1")
sed -n -e '/regexp/{=;x;1!p;g;$!N;p;D;}' -e h

# grep for AAA and BBB and CCC (in any order)
sed '/AAA/!d; /BBB/!d; /CCC/!d'

# grep for AAA and BBB and CCC (in that order)
sed '/AAA.*BBB.*CCC/!d'

# grep for AAA or BBB or CCC (emulates "egrep")
sed -e '/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d # most seds
gsed '/AAA\|BBB\|CCC/!d'                      # GNU sed only

# print paragraph if it contains AAA (blank lines separate paragraphs)
# HHsed v1.5 must insert a 'G;' after 'x;' in the next 3 scripts below
sed -e '/./{H;$!d;}' -e 'x;/AAA/!d;'

# print paragraph if it contains AAA and BBB and CCC (in any order)
sed -e '/./{H;$!d;}' -e 'x;/AAA/!d;/BBB/!d;/CCC/!d'

# print paragraph if it contains AAA or BBB or CCC
sed -e '/./{H;$!d;}' -e 'x;/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d
gsed '/./{H;$!d;};x;/AAA\|BBB\|CCC/b;d'        # GNU sed only

# print only lines of 65 characters or longer
sed -n '/^.\{65\}/p'

# print only lines of less than 65 characters
sed -n '/^.\{65\}/!p'        # method 1, corresponds to above
sed '/^.\{65\}/d'            # method 2, simpler syntax

# print section of file from regular expression to end of file
sed -n '/regexp/,$p'

# print section of file based on line numbers (lines 8-12, inclusive)
sed -n '8,12p'               # method 1
sed '8,12!d'                 # method 2

# print line number 52
sed -n '52p'                 # method 1
sed '52!d'                   # method 2
sed '52q;d'                # method 3, efficient on large files

# beginning at line 3, print every 7th line
gsed -n '3~7p'             # GNU sed only
sed -n '3,${p;n;n;n;n;n;n;}' # other seds

# print section of file between two regular expressions (inclusive)
sed -n '/Iowa/,/Montana/p'           # case sensitive
```

SELECTIVE DELETION OF CERTAIN LINES:

```
# print all of file EXCEPT section between 2 regular expressions
sed '/Iowa/,/Montana/d'

# delete duplicate, consecutive lines from a file (emulates "uniq").
# First line in a set of duplicate lines is kept, rest are deleted.
sed '$!N; /^\(.*\)\n\1$/!P; D'

# delete duplicate, nonconsecutive lines from a file. Beware not to
# overflow the buffer size of the hold space, or else use GNU sed.
sed -n 'G; s/\n/&&/; /^\([ -~]*\n\).*\n\1/d; s/\n//; h; P'

# delete all lines except duplicate lines (emulates "uniq -d").
sed '$!N; s/^\(.*\)\n\1$/\1/; t; D'

# delete the first 10 lines of a file
sed '1,10d'

# delete the last line of a file
sed '$d'

# delete the last 2 lines of a file
sed 'N;$!P;$!D;$d'

# delete the last 10 lines of a file
sed -e :a -e '$d;N;2,10ba' -e 'P;D' # method 1
sed -n -e :a -e '1,10!{P;N;D;};N;ba' # method 2

# delete every 8th line
gsed '0~8d'                    # GNU sed only
sed 'n;n;n;n;n;n;n;d;'              # other seds

# delete lines matching pattern
sed '/pattern/d'

# delete ALL blank lines from a file (same as "grep '.' ")
sed '/^$/d'                     # method 1
sed '/./!d'                     # method 2
```

```
# delete all CONSECUTIVE blank lines from file except the first; also
# deletes all blank lines from top and end of file (emulates "cat -s")
sed '/./,/^$/!d'         # method 1, allows 0 blanks at top, 1 at EOF
sed '/^$/N;/\n$/D'       # method 2, allows 1 blank at top, 0 at EOF

# delete all CONSECUTIVE blank lines from file except the first 2:
sed '/^$/N;/\n$/N;//D'

# delete all leading blank lines at top of file
sed '/./,$!d'

# delete all trailing blank lines at end of file
sed -e :a -e '/^\n*$/{$d;N;ba' -e '}' # works on all seds
sed -e :a -e '/^\n*$/N;/\n$/ba'       # ditto, except for gsed 3.02.*

# delete the last line of each paragraph
sed -n '/^$/{p;h;};/./{x;/./p;}'
```

SPECIAL APPLICATIONS:

```
# remove nroff overstrikes (char, backspace) from man pages. The 'echo'
# command may need an -e switch if you use Unix System V or bash shell.
sed "s/.`echo \\\b`//g" # double quotes required for Unix environment
sed 's/.^H//g'           # in bash/tcsh, press Ctrl-V and then Ctrl-H
sed 's/.\x08//g'         # hex expression for sed 1.5, GNU sed, ssed

# get Usenet/e-mail message header
sed '/^$/q'              # deletes everything after first blank line

# get Usenet/e-mail message body
sed '1,/^$/d'            # deletes everything up to first blank line

# get Subject header, but remove initial "Subject: " portion
sed '/^Subject: */!d; s///;q'

# get return address header
sed '/^Reply-To:/q; /^From:/h; /./d;g;q'

# parse out the address proper. Pulls out the e-mail address by itself
# from the 1-line return address header (see preceding script)
sed 's/ *(.*)//; s/>.*//; s/.*[:<] *//'

# add a leading angle bracket and space to each line (quote a message)
sed 's/^/> /'

# delete leading angle bracket & space from each line (unquote a message)
sed 's/^> //'
```

```
 # remove most HTML tags (accommodates multiple-line tags)
 sed -e :a -e 's/<[^>]*>//g;/</N;//ba'

 # extract multi-part uuencoded binaries, removing extraneous header
 # info, so that only the uuencoded portion remains. Files passed to
 # sed must be passed in the proper order. Version 1 can be entered
 # from the command line; version 2 can be made into an executable
 # Unix shell script. (Modified from a script by Rahul Dhesi.)
 sed '/^end/,/^begin/d' file1 file2 ... fileX | uudecode # vers. 1
 sed '/^end/,/^begin/d' "$@" | uudecode                  # vers. 2

 # sort paragraphs of file alphabetically. Paragraphs are separated by blank
 # lines. GNU sed uses \v for vertical tab, or any unique char will do.
 sed '/./{H;d;};x;s/\n/={NL}=/g' file | sort | sed '1s/={NL}=//;s/={NL}=/\n/g'
 gsed '/./{H;d};x;y/\n/\v/' file | sort | sed '1s/\v//;y/\v/\n/'

 # zip up each .TXT file individually, deleting the source file and
 # setting the name of each .ZIP file to the basename of the .TXT file
 # (under DOS: the "dir /b" switch returns bare filenames in all caps).
 echo @echo off >zipup.bat
 dir /b *.txt | sed "s/^\(.*\)\.TXT/pkzip -mo \1 \1.TXT/" >>zipup.bat
```

TYPICAL USE: Sed takes one or more editing commands and applies all of them, in sequence, to each line of input. After all the commands have been applied to the first input line, that line is output and a second input line is taken for processing, and the cycle repeats. The preceding examples assume that input comes from the standard input device (i.e, the console, normally this will be piped input). One or more filenames can be appended to the command line if the input does not come from stdin. Output is sent to stdout (the screen). Thus:

```
 cat filename | sed '10q' # uses piped input
 sed '10q' filename          # same effect, avoids a useless "cat"
 sed '10q' filename > newfile # redirects output to disk
```

For additional syntax instructions, including the way to apply editing commands from a disk file instead of the command line, consult "sed & awk, 2nd Edition," by Dale Dougherty and Arnold Robbins (O'Reilly, 1997; http://www.ora.com), "UNIX Text Processing," by Dale Dougherty and Tim O'Reilly (Hayden Books, 1987) or the tutorials by Mike Arst distributed in U-SEDIT2.ZIP (many sites). To fully exploit the power of sed, one must understand "regular expressions." For this, see "Mastering Regular Expressions" by Jeffrey Friedl (O'Reilly, 1997). The manual ("man") pages on Unix systems may be helpful (try "man sed", "man regexp", or the subsection on regular expressions in "man ed"), but man pages are notoriously difficult. They are not written to teach sed use or regexps to first-time users, but as a reference text for those already acquainted with these tools.

QUOTING SYNTAX: The preceding examples use single quotes ('...') instead of double quotes ("...") to enclose editing commands, since sed is typically used on a Unix platform. Single quotes prevent the Unix shell from intrepreting the dollar sign ($) and backquotes (`...`), which are expanded by the shell if they are enclosed in double quotes. Users of the "csh" shell and derivatives will also need to quote the exclamation mark (!) with the backslash (i.e., \!) to properly run the examples listed above, even within single quotes. Versions of sed written for DOS invariably require double quotes ("...") instead of single quotes to enclose editing commands.

USE OF '\t' IN SED SCRIPTS: For clarity in documentation, we have used the expression '\t' to indicate a tab character (0x09) in the scripts. However, most versions of sed do not recognize the '\t' abbreviation, so when typing these scripts from the command line, you should press the TAB key instead. '\t' is supported as a regular expression metacharacter in awk, perl, and HHsed, sedmod, and GNU sed v3.02.80.

VERSIONS OF SED: Versions of sed do differ, and some slight syntax variation is to be expected. In particular, most do not support the use of labels (:name) or branch instructions (b,t) within editing commands, except at the end of those commands. We have used the syntax which will be portable to most users of sed, even though the popular GNU versions of sed allow a more succinct syntax. When the reader sees a fairly long command such as this:

```
  sed -e '/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d
```

it is heartening to know that GNU sed will let you reduce it to:

```
  sed '/AAA/b;/BBB/b;/CCC/b;d'      # or even
  sed '/AAA\|BBB\|CCC/b;d'
```

In addition, remember that while many versions of sed accept a command like "/one/ s/RE1/RE2/", some do NOT allow "/one/! s/RE1/RE2/", which contains space before the 's'. Omit the space when typing the command.

OPTIMIZING FOR SPEED: If execution speed needs to be increased (due to large input files or slow processors or hard disks), substitution will be executed more quickly if the "find" expression is specified before giving the "s/.../.../" instruction. Thus:

```
  sed 's/foo/bar/g' filename         # standard replace command
  sed '/foo/ s/foo/bar/g' filename # executes more quickly
  sed '/foo/ s//bar/g' filename      # shorthand sed syntax
```

On line selection or deletion in which you only need to output lines
from the first part of the file, a "quit" command (q) in the script
will drastically reduce processing time for large files. Thus:

```
  sed -n '45,50p' filename          # print line nos. 45-50 of a file
  sed -n '51q;45,50p' filename      # same, but executes much faster
```

If you have any additional scripts to contribute or if you find errors
in this document, please send e-mail to the compiler. Indicate the
version of sed you used, the operating system it was compiled for, and
the nature of the problem. To qualify as a one-liner, the command line
must be 65 characters or less. Various scripts in this file have been
written or contributed by:

```
 Al Aab                 # founder of "seders" list
 Edgar Allen            # various
 Yiorgos Adamopoulos    # various
 Dale Dougherty         # author of "sed & awk"
 Carlos Duarte          # author of "do it with sed"
 Eric Pement            # author of this document
 Ken Pizzini            # author of GNU sed v3.02
 S.G. Ravenhall         # great de-html script
 Greg Ubben             # many contributions & much help
```

-------------------------------------------------------

-------------------------------------------------------------------------------------------

HANDY ONE-LINERS FOR SED (Unix stream editor) Oct. 29, 1997
compiled by Eric Pement <epement@jpusa.chi.il.us> version 4.3
Latest version of this file is always at <http://www.wollery.demon.co.uk>
FILE SPACING:

```
# double space a file
sed G

# triple space a file
sed 'G;G'

# undo double-spacing (assumes even-numbered lines are always blank)
sed 'n;d'
```

NUMBERING:

```
# number each line of a file (simple left alignment). Using a tab (see
# note on '\t' at end of file) instead of space will preserve margins.
sed = filename | sed 'N;s/\n/\t/'

# number each line of a file (number on left, right-aligned)
sed = filename | sed 'N; s/^/ /; s/ *\(.\{6,\}\)\n/\1 /'
```

```
# number each line of file, but only print numbers if line is not blank
sed '/./=' filename | sed '/./N; s/\n/ /'

# count lines (emulates "wc -l")
sed -n '$='
```

TEXT CONVERSION AND SUBSTITUTION:

```
# IN UNIX ENVIRONMENT: convert DOS newlines (CR/LF) to Unix format
sed 's/.$//'

# IN DOS ENVIRONMENT: convert Unix newlines (LF) to DOS format
sed 's/$//' # method 1
sed -n p # method 2

# delete leading whitespace (spaces, tabs) from front of each line
# aligns all text flush left
sed 's/^[ \t]*//' # see note on '\t' at end of file

# delete trailing whitespace (spaces, tabs) from end of each line
sed 's/[ \t]*$//' # see note on '\t' at end of file

# delete BOTH leading and trailing whitespace from each line
sed 's/^[ \t]*//;s/[ \t]*$//'

# insert 5 blank spaces at beginning of each line (make page offset)
sed 's/^/ /'

# align all text flush right on a 79-column width
sed -e :a -e 's/^.\{1,78\}$/ &/;ta' # set at 78 plus 1 space

# center all text in the middle of 79-column width. In method 1,
# spaces at the beginning of the line are significant, and trailing
# spaces are appended at the end of the line. In method 2, spaces at
# the beginning of the line are discarded in centering the line, and
# no trailing spaces appear at the end of lines.
sed -e :a -e 's/^.\{1,77\}$/ & /;ta' # method 1
sed -e :a -e 's/^.\{1,77\}$/ &/;ta' -e 's/\( *\)\1/\1/' # method 2

# substitute (find & replace) "foo" with "bar" on each line
sed 's/foo/bar/' # replaces only 1st instance in a line
sed 's/foo/bar/4' # replaces only 4th instance in a line
sed 's/foo/bar/g' # replaces ALL instances in a line

# substitute "foo" with "bar" ONLY for lines which contain "baz"
sed '/baz/s/foo/bar/g'

# substitute "foo" with "bar" EXCEPT for lines which contain "baz"
```

```
sed '/baz/!s/foo/bar/g'

# reverse order of lines (emulates "tac")
sed '1!G;h;$!d'

# reverse each character on the line (emulates "rev")
sed '/\n/!G;s/\(.\)\(.*\n\)/&\2\1/;//D;s/.//'

# join pairs of lines side-by-side (like "paste")
sed 'N;s/\n/ /'
```

SELECTIVE PRINTING OF CERTAIN LINES:

```
# print first 10 lines of file (emulates behavior of "head")
sed 10q

# print first line of file (emulates "head -1")
sed q

# print last 10 lines of file (emulates "tail")
sed -e :a -e '$q;N;11,$D;ba'

# print last line of file (emulates "tail -1")
sed '$!d'

# print only lines which match regular expression (emulates "grep")
sed -n '/regexp/p' # method 1
sed '/regexp/!d' # method 2

# print only lines which do NOT match regexp (emulates "grep -v")
sed -n '/regexp/!p' # method 1, corresponds to above
sed '/regexp/d' # method 2, simpler syntax

# print 1 line of context before and after regexp, with line number
# indicating where the regexp occurred (similar to "grep -A1 -B1")
sed -n -e '/regexp/{=;x;1!p;g;$!N;p;D;}' -e h

# grep for AAA and BBB and CCC (in any order)
sed '/AAA/!d; /BBB/!d; /CCC/!d'

# grep for AAA or BBB or CCC (emulates "egrep")
sed -e '/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d

# print only lines of 65 characters or longer
sed -n '/^.\{65\}/p'

# print only lines of less than 65 characters
sed -n '/^.\{65\}/!p' # method 1, corresponds to above
sed '/^.\{65\}/d' # method 2, simpler syntax

# print section of file from regular expression to end of file
sed -n '/regexp/,$p'

# print section of file based on line numbers (lines 8-12, inclusive)
sed -n '8,12p' # method 1
sed '8,12!d' # method 2

# print line number 52
sed -n '52p' # method 1
sed '52!d' # method 2
sed '52q;d' # method 3, efficient on large files

# print section of file between two regular expressions (inclusive)
sed -n '/Iowa/,/Montana/p' # case sensitive
```

SELECTIVE DELETION OF CERTAIN LINES:

```
# print all of file EXCEPT section between 2 regular expressions
sed '/Iowa/,/Montana/d'

# delete duplicate lines from a sorted file (emulates "uniq"). First
# line in a set of duplicate lines is kept, the rest are deleted
sed '$!N; /^\(.*\)\n\1$/!P; D'

# delete ALL blank lines from a file (same as "grep '.' ")
sed '/^$/d'

# delete all CONSECUTIVE blank lines from file except the first; also
# deletes all blank lines from top and end of file (emulates "cat -s")
sed '/./,/^$/!d' # method 1, allows 0 blanks at top, 1 at EOF
sed '/^$/N;/\n$/D' # method 2, allows 1 blank at top, 0 at EOF

# delete all CONSECUTIVE blank lines from file except the first 2:
sed '/^$/N;/\n$/N;//D'

# delete all leading blank lines at top of file
sed '/./,$!d'

# delete all trailing blank lines at end of file
sed -e :a -e '/^\n*$/N;/\n$/ba'
```

SPECIAL APPLICATIONS:

```
# remove nroff overstrikes (char, backspace) from man pages
sed "s/.`echo \\\b`//g" # double quotes required for Unix environment
sed 's/.\x08//g' # hex expression for GNU sed (octal is "\010")
```

```
# get Usenet/e-mail message header
sed '/^$/q' # deletes everything after first blank line

# get Usenet/e-mail message body
sed '1,/^$/d' # deletes everything up to first blank line

# get Subject header, but remove initial "Subject: " portion
sed '/^Subject: */!d; s///;q'

# get return address header
sed '/^Reply-To:/q; /^From:/h; /./d;g;q'

# parse out the address proper. Pulls out the e-mail address by itself
# from the 1-line return address header (see preceding script)
sed 's/ *(.*)//; s/>.*//; s/.*[:<] *//'

# add a leading angle bracket and space to each line (quote a message)
sed 's/^/> /

# delete leading angle bracket & space from each line (unquote a message)
sed 's/^> //'

# remove most HTML tags (accommodates multiple-line tags)
sed -e :a -e 's/<[^<]*>/ /g;/</{N;s/\n/ /;ba;}'

# extract multi-part uuencoded binaries, removing extraneous header
# info, so that only the uuencoded portion remains. Files passed to
# sed must be passed in the proper order. Version 1 can be entered
# from the command line; version 2 can be made into an executable
# Unix shell script. (Modified from a script by Rahul Dhesi.)
sed '/^end/,/^begin/d' file1 file2 ... fileX | uudecode # vers. 1
sed '/^end/,/^begin/d' $* | uudecode # vers. 2

# zip up each .TXT file individually, deleting the source file and
# setting the name of each .ZIP file to the basename of the .TXT file
# (under DOS: the "dir /b" switch returns bare filenames in all caps).
echo @echo off >zipup.bat
dir /b *.txt | sed "s/^\(.*\)\.TXT/pkzip -mo \1 \1.TXT/" >>zipup.bat
```

TYPICAL USE: Sed takes one or more editing commands and applies all of them, in sequence, to each line of input. After all the commands have been applied to the first input line, that line is output and a second input line is taken for processing, and the cycle repeats. The preceding examples assume that input comes from the standard input device (i.e, the console, normally this will be piped input). One or more filenames can be appended to the command line if the input does not come from stdin. Output is sent to stdout (the screen). Thus:

```
cat filename | sed '10q' # uses piped input
sed '10q' filename # same effect, avoids a useless "cat"
sed '10q' filename > newfile # redirects output to disk
```

For additional syntax instructions, including the way to apply editing commands from a disk file instead of the command line, consult "sed & awk, 2nd Edition," by Dale Dougherty and Arnold Robbins (O'Reilly, 1997; http://www.ora.com), "UNIX Text Processing," by Dale Dougherty and Tim O'Reilly (Hayden Books, 1987) or the tutorials by Mike Arst distributed in U-SEDIT2.ZIP (many sites). To fully exploit the power of sed, one must understand "regular expressions." For this, see "Mastering Regular Expressions" by Jeffrey Friedl (O'Reilly, 1997). The manual ("man") pages on Unix systems may be helpful (try "man sed", "man regexp", or the subsection on regular expressions in "man ed"), but man pages are notoriously difficult. They are not written to teach sed use or regexps to first-time users, but as a reference text for those already acquainted with these tools.

QUOTING SYNTAX: The preceding examples use single quotes ('...') instead of double quotes ("...") to enclose editing commands, since sed is typically used on a Unix platform. Single quotes prevent the Unix shell from intrepreting the dollar sign ($) and backquotes (`...`), which are expanded by the shell if they are enclosed in double quotes. Users of the "csh" shell and derivatives will also need to quote the exclamation mark (!) with the backslash (i.e., \!) to properly run the examples listed above, even within single quotes. Versions of sed written for DOS invariably require double quotes ("...") instead of single quotes to enclose editing commands.

USE OF '\t' IN SED SCRIPTS: For clarity in documentation, we have used the expression '\t' to indicate a tab character (0x09) in the scripts. However, most versions of sed do not recognize the '\t' abbreviation, so when typing these scripts from the command line, you should press the TAB key instead. '\t' is supported as a regular expression metacharacter in awk, perl, and in a few implementations of sed.

VERSIONS OF SED: Versions of sed do differ, and some slight syntax variation is to be expected. In particular, most do not support the use of labels (:name) or branch instructions (b,t) within editing commands, except at the end of those commands. We have used the syntax which will be portable to most users of sed, even though the popular GNU versions of sed allow a more succinct syntax. When the reader sees a fairly long command such as this:

```
sed -e '/AAA/b' -e '/BBB/b' -e '/CCC/b' -e d
```

it is heartening to know that GNU sed will let you reduce it to:

```
sed '/AAA/b;/BBB/b;/CCC/b;d'
```

In addition, remember that while many versions of sed accept a command like "/one/ s/RE1/RE2/", some do NOT allow "/one/! s/RE1/RE2/", which contains space before the 's'. Omit the space when typing the command.

OPTIMIZING FOR SPEED: If execution speed needs to be increased (due to large input files or slow processors or hard disks), substitution will be executed more quickly if the "find" expression is specified before giving the "s/.../.../" instruction. Thus:

```
sed 's/foo/bar/g' filename # standard replace command
sed '/foo/ s/foo/bar/g' filename # executes more quickly
sed '/foo/ s//bar/g' filename # shorthand sed syntax
```

On line selection or deletion in which you only need to output lines from the first part of the file, a "quit" command (q) in the script will drastically reduce processing time for large files. Thus:

```
sed -n '45,50p' filename # print line nos. 45-50 of a file
sed -n '51q;45,50p' filename # same, but executes much faster
```

If you have any additional scripts to contribute or if you find errors in this document, please send e-mail to the compiler. Indicate the version of sed you used, the operating system it was compiled for, and the nature of the problem. Various scripts in this file were written or contributed by:

## Answers

1. To display lines from x1 having exactly three characters.
-> $grep '^...$' x1

2. To display lines from file x1 that begins with any alphabets.
-> $grep '^[a-zA-Z]' x1

3. To display lines from file x1 that having the string “unix” at least twice.
-> $grep 'unix.* unix' x1

4. To display non blank lines from file x1.
-> $grep '.' x1

5. To display lines having exactly 50 characters of file x1.
-> $grep -n 50 x1

6. Count number of blank lines in file x1.

-> $grep -c -v '.' x1

7. To display lines having at least one * character in file x1.
-> $grep *'/*'* x1

8. To display lines from file x1 that containing string “UNIX” or “Unix” or “unix”.
-> $grep -i 'unix' x1

Answers

1. To finds the frequency of each word in a file.
-> $awk

2. To display record which having more than 100 characters excluding white space characters.
-> $awk 'length c 100'

3. To display records having exactly five fields.
-> $awk -F "1" 'NF==5' x1

4. To display all non-blank lines along with the line numbers.
-> $awk -F "1" length !{print NR}{print}

5. To display the length of the first field of every line of file x1.
-> $awk -F "1" {print length ($1)} x1

6. To display lines containing a string UNIX in the second file of x1.
-> $awk -F "1" $d="UNIX" x1

7. To display last line having odd number of fields of file x1.
-> $awk

8. To display number of words and lines in given file x1.
-> $awk

9. To display only first time having odd number of fields of file x1.
-> $awk 'NF%2!=0' {count=count+1;if(count=0)print} x1

10. To display the length of the last field of every line of file x1.
-> $awk -F "1" '{print length($NF)}'

11. To display lines having at least three fields from file x1.
-> $awk -F "1" 'NF>=3' x1

12. To display fields of each line in reverse order of file x1.
-> $awk '{for(i=NF;i>0;i--)print i;}'

13. To display only those records having even number of fields.

-> $awk -F "1" 'NF%2==0' x1

14. To display total number of bytes of occupied by files available in working directory.
-> $awk

15. To display all user id having super user privileges. If 3rd field of /etc/passwd is 0 then user having super user privileges further field separator is colon (:).
-> $awk

16. To display a record number having maximum of fields.
-> $awk -F "1" '{print NF}' x1

17. To display number of fields.
-> $awk -F "1" '{print NF}' x1

18. To display total number of fields for each record.
-> $awk -F "1" '{if(substr($1,1,1)~/[a-zA-Z]/)print}' x1|wc -l

19. To display total number of records whose 1st field begins with small or capital alphabets.
-> $awk