Regular Expressions in Unix/Linux/Cygwin

CS 162 – UC-Irvine

Some slides from Reva Freedman, Marty Stepp, Jessica Miller, and Ruth Anderson

Regular Expression (RE) Formal Definition

- Basis:
  - single character, a, is an RE, signifying language {a}.
  - ε is an RE, signifying language $\{\varepsilon\}$
  - $\emptyset$ is an RE, signifying language $\emptyset$
- If E_1 and E_2 are REs, then $E_1|E_2$ is an RE, signifying $L(E_1) \cup L(E_2)$
- If E_1 and E_2 are REs, then E_1E_2 is an RE, signifying $L(E_1)\ L(E_2)$, that is, concatenation
- If E is an RE, then E* is an RE, signifying L(E)*, that is, Kleene closure, which is the concatenation of 0 or more strings from L(E).
- Precedence is the the order of Kleene closure (highest), concatenation, and union (lowest)
- Parentheses can be used for grouping and don't count as characters.

`egrep` and regexes

command	description
`egrep`	extended `grep`; uses regexes in its search patterns; equivalent to `grep -E`

```
egrep "[0-9]{3}-[0-9]{3}-[0-9]{4}" contact.html
```

- `egrep` searches for a regular expression pattern in a file (or group of files)
- `grep` uses “basic” regular expressions instead of “extended”
  - extended has some minor differences and additional metacharacters
- `-i` option before regex signifies a case-insensitive match
  - `egrep -i "mart"` matches "Marty S", "smartie", "WALMART", ...

Metacharacters

RE Metacharacter	Matches…
.	**Any one character, except new line**
[a-z]	**Any one of the enclosed characters (e.g. a-z)**
*	**Zero or more of preceding character**
? or \?	**Zero or one of the preceding characters**
+ or \+	**One or more of the preceding characters**

- any non-metacharacter matches itself

more Metacharacters

RE Metacharacter	Matches…
^	**beginning of line**
$	**end of line**
char	**Escape the meaning of *char* following it**
[^]	**One character not in the set**
\<	**Beginning of word anchor**
\>	**End of word anchor**
( ) or \(\)	**Tags matched characters to be used later (max = 9)**
| or |	**Or grouping**
x\{m\}	**Repetition of character x, m times (x,m = integer)**
x\{m,\}	**Repetition of character x, at least m times**
x\{m,n\}	**Repetition of character x between m and m times**

Wildcards and anchors

`.` (a dot) matches any character except `\n`

- `".oo.y"` matches `"Doocy"`, `"goofy"`, `"LooPy"`, …
- use `\.` to literally match a dot `.` character

`^` matches the beginning of a line; `$` the end

- `"^fi$"` matches lines that consist entirely of `fi`

`\<` demands that pattern is the beginning of a *word*;
`\>` demands that pattern is the end of a word

- `"\<for\>"` matches lines that contain the word `"for“`
- Words are made up of letters, digits and _ (underscore)

Special characters

| means OR

- `"abc|def|g"` matches lines with "abc", "def", or "g"
- precedence of `^(Subject|Date)` vs. `^Subject|Date:`
- There's no AND symbol.

( ) are for grouping

- `"(Homer|Marge) Simpson"` matches lines containing `"Homer Simpson"` or `"Marge Simpson"`

\ starts an escape sequence

- many characters must be escaped to match them: `/ \ $ . [ ] ( ) ^ * + ?`
- `"\.\\n"` matches lines containing `".\n"`

Quantifiers: * + ?

`*` means 0 or more occurrences

- "ab`c*`" matches "ab", "abc", "abcc", "abccc", ...
- "a`(bc)*`" matches "a", "abc", "abcbc", "abcbcbc", ...
- "a`.*`a" matches "aa", "aba", "a8qa", "a!?_a", ...

`+` means 1 or more occurrences

- "a`(bc)+`" matches "abc", "abcbc", "abcbcbc", ...
- "Go`o+`gle" matches "Google", "Gooogle", "Goooogle", ...

`?` means 0 or 1 occurrences

- "Martin`a?`" matches lines with "Martin" or "Martina"
- "Dan`(iel)?`" matches lines with "Dan" or "Daniel"

More quantifiers

`{`***`min`***`,`***`max`***`}` means between ***min*** and ***max*** occurrences

- `"a(bc){2,4}"` matches "abcbc", "abcbcbc", or "abcbcbcbc"

- ***min*** or ***max*** may be omitted to specify any number
  - `"{2,}"` means 2 or more
  - `"{,6}"` means up to 6
  - `"{3}"` means exactly 3

Character sets

`[ ]` group characters into a character set;
will match any single character from the set

- `"[bcd]art"` matches strings containing "bart", "cart", and "dart"
- equivalent to `"(b|c|d)art"` but shorter

- inside `[ ]`, most modifier keys act as normal characters
  - `"what[.!*?]*"` matches "what", "what.", "what!", "what?**!", ...

Character ranges

- inside a character set, specify a range of characters with -
  - `"[a-z]"` matches any lowercase letter
  - `"[a-zA-Z0-9]"` matches any lower- or uppercase letter or digit

- an initial ^ inside a character set **negates** it
  - `"[^abcd]"` matches any character **other than** a, b, c, or d

- inside a character set, - can sometimes be tricky to match
  - Try escaping it (use \) or place it last in the brackets
  - `"[+\-]?[0-9]+"` matches optional + or -, followed by ≥ one digit

POSIX Character Sets

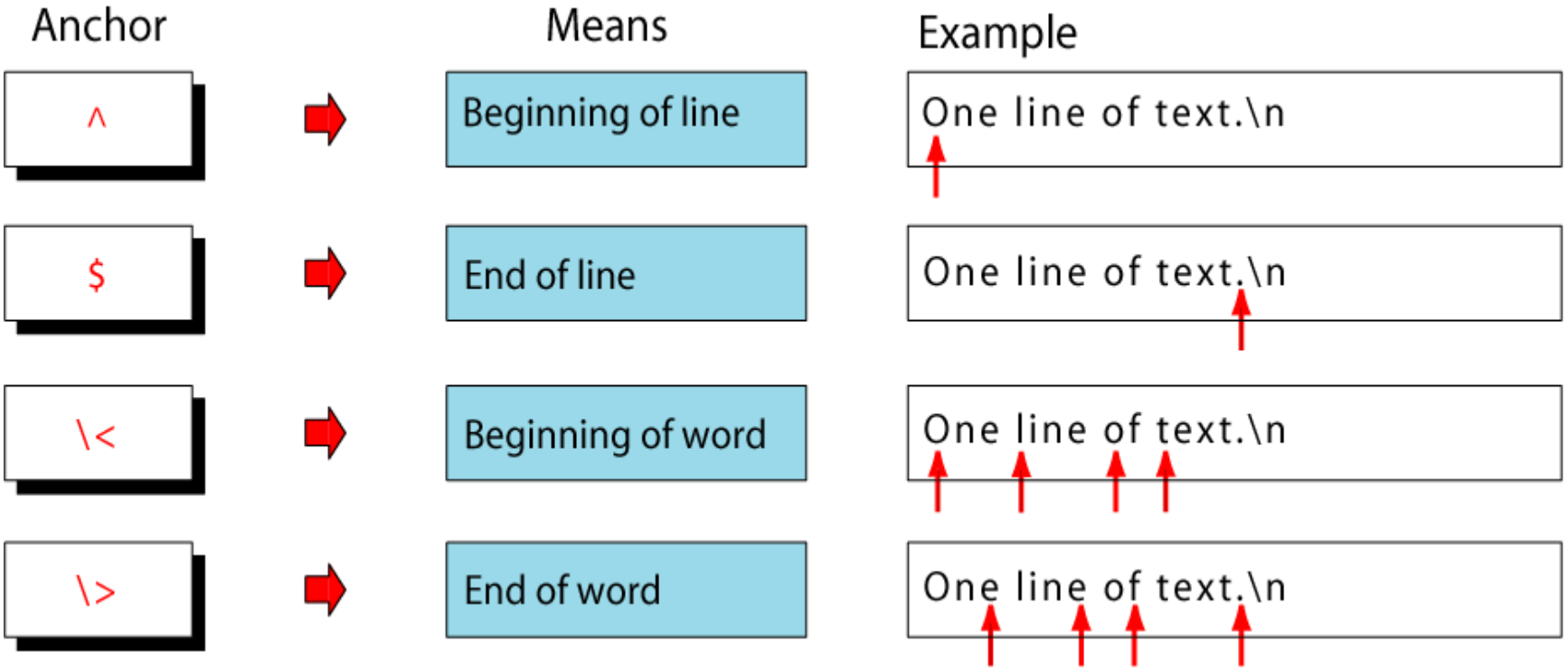
- POSIX added newer, portable ways to describe character sets:

Character Group	Meaning
[:alnum:]	Alphanumeric
[:cntrl:]	Control Character
[:lower:]	Lower case character
[:space:]	Whitespace
[:alpha:]	Alphabetic
[:digit:]	Digit
[:print:]	Printable character
[:upper:]	Upper Case Character
[:blank:]	whitespace, tabs, etc.
[:graph:]	Printable and visible characters
[:punct:]	Punctuation
[:xdigit:]	Extended Digit

- Note that some people use [[:alpha:]] as a notation, but the outer '[...]' specifies a character set.

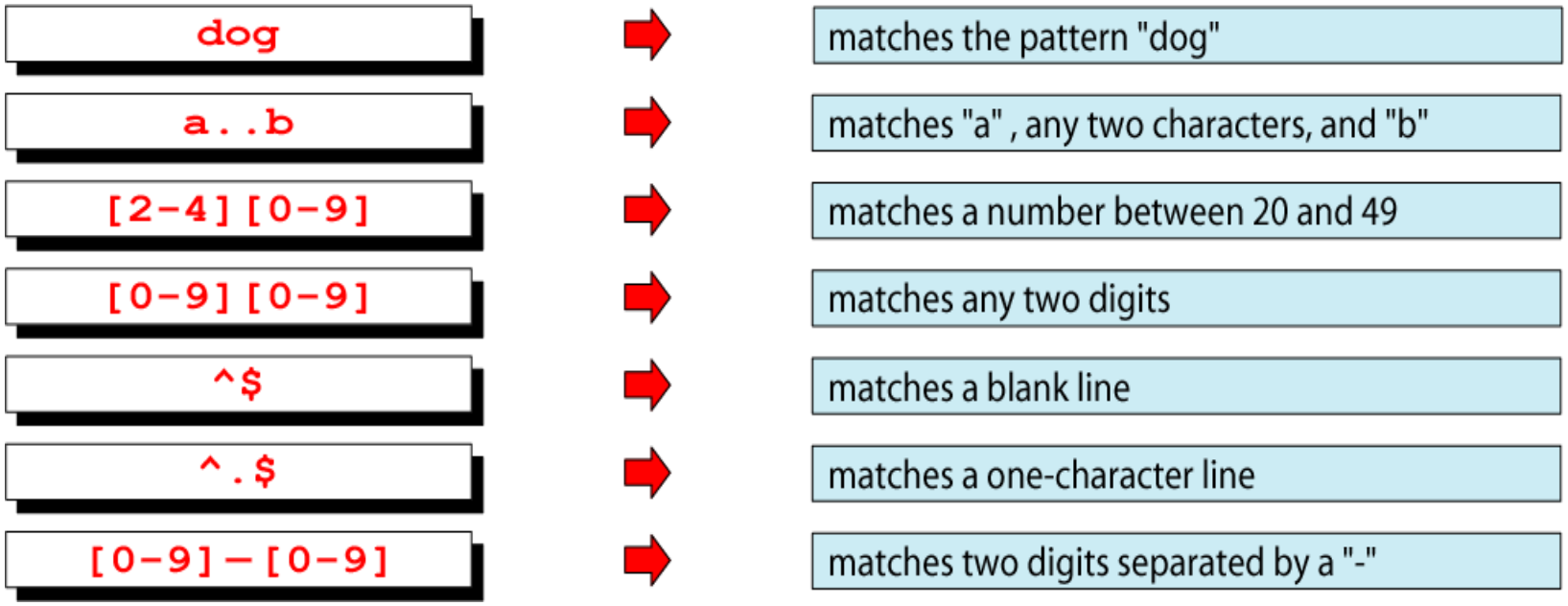
Anchors

Anchors tell where the next character in the pattern must be located in the text data.



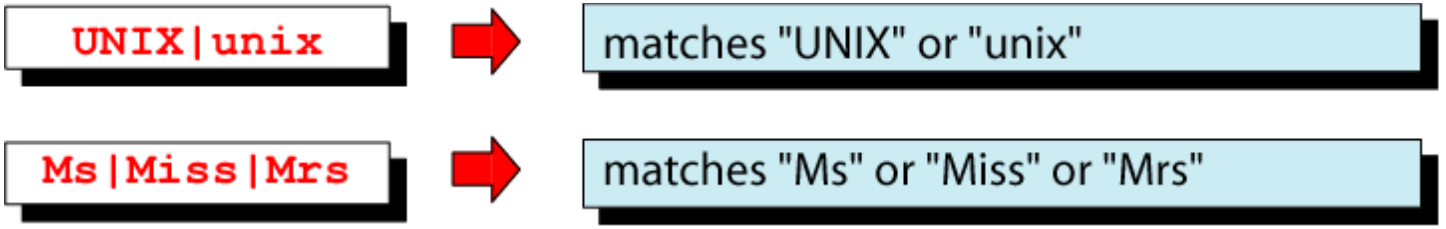
Concatenation Operator

In a sequence operator, if a series of atoms are shown in a regular expression, there is no operator between them.



Alternation Operator: | or \|

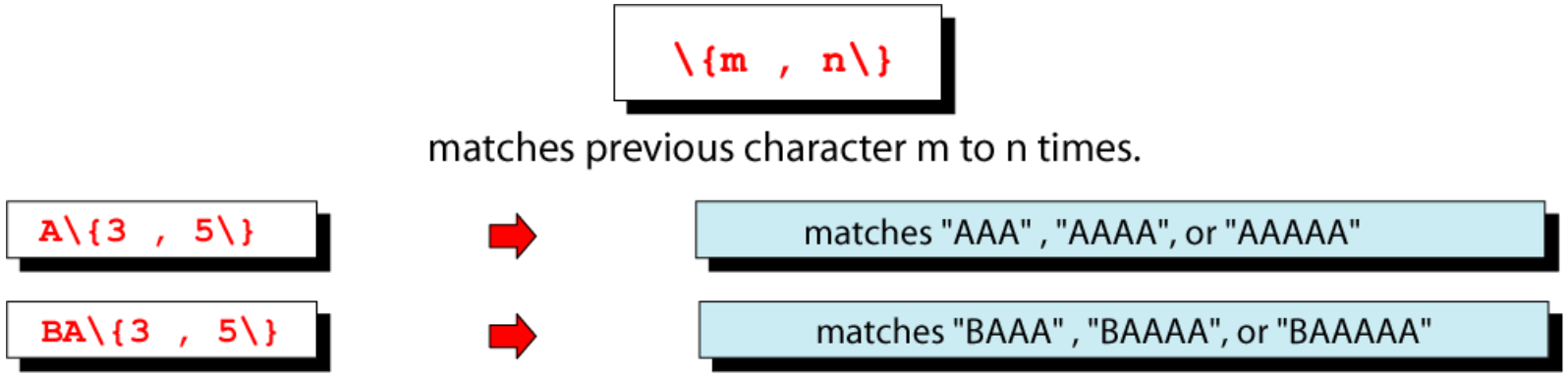
operator (| or \|) is used to define one
or more alternatives



Note: depends on whether using “egrep” or “grep”

Repetition Operator: {…} or \{…\}

The repetition operator specifies that the atom or expression immediately before the repetition may be repeated.



Basic Repetition Forms

Formats

`\{m\}`	➡	matches previous atom exactly m times
`\{m, \}`	➡	matches previous atom m times or more
`\{, n\}`	➡	matches previous atom n times or less

Examples

`CA\{5\}`	➡	`CAAAAA`
`CA\{3,\}`	➡	`CAAA, CAAAA, CAAAAA, …`
`CA\{,2\}`	➡	`C, CA, CAA`

Short Form Repetition Operators: * + ?

Formats

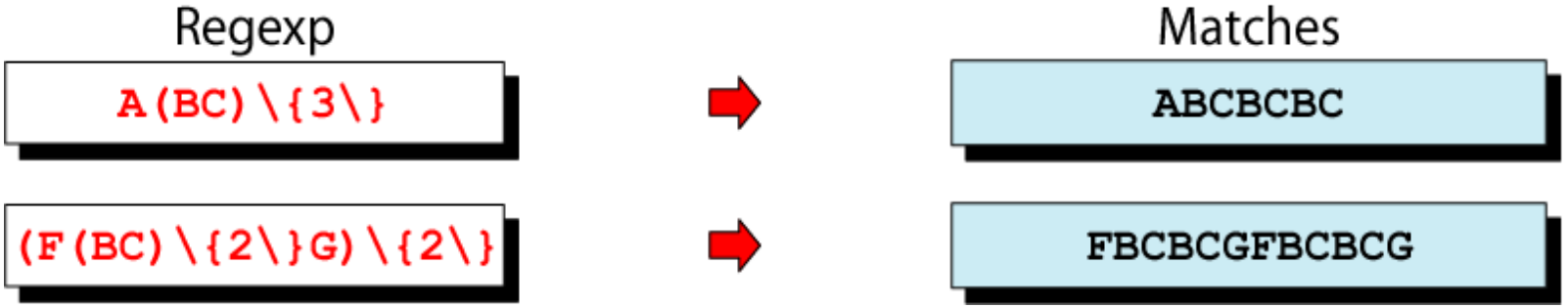
`*`	➡	special case: matches previous atom zero or more times
`+`	➡	special case: matches previous atom one or more times
`?`	➡	special case: matches previous atom 0 or one time only

Examples

`BA*`	➡	B, BA, BAA, BAAA, BAAAA, . . .
`B.*`	➡	B, BA . . . BZ, BAA . . . BZZ, BAAA . . . BZZZ, . . .
`.*`	➡	zero or more characters
`.+`	➡	one or more characters
`[0-9]?`	➡	zero or one digit

Group Operator

In the group operator, when a group of characters is enclosed in parentheses, the next operator applies to the whole group, not only the previous characters.



Note: depends on “egrep” or “grep”
- grep uses \(and \)

Grep detail and examples

- grep is family of commands
  - grep (global regular expression print)
    common version
  - egrep (extended grep)
    understands extended REs
    (| + ? ( ) don’t need backslash)
  - fgrep (fast grep)
    understands only fixed strings, i.e., is faster

Commonly used “grep” options:

-c	Print only a count of matched lines.
-i	Ignore uppercase and lowercase distinctions.
-l	List all files that contain the specified pattern.
-n	Print matched lines and line numbers.
-s	Work silently; display nothing except error messages. Useful for checking the exit status.
-v	Print lines that do not match the pattern.

Example: grep with pipe

Pipe the output of the "ls –l" command to grep and list/select only directory entries.

```
% ls -l | grep '^d'
drwxr-xr-x   2 krush    csci        512 Feb  8 22:12 assignments
drwxr-xr-x   2 krush    csci        512 Feb  5 07:43 feb3
drwxr-xr-x   2 krush    csci        512 Feb  5 14:48 feb5
drwxr-xr-x   2 krush    csci        512 Dec 18 14:29 grades
drwxr-xr-x   2 krush    csci        512 Jan 18 13:41 jan13
drwxr-xr-x   2 krush    csci        512 Jan 18 13:17 jan15
drwxr-xr-x   2 krush    csci        512 Jan 18 13:43 jan20
drwxr-xr-x   2 krush    csci        512 Jan 24 19:37 jan22
drwxr-xr-x   4 krush    csci        512 Jan 30 17:00 jan27
drwxr-xr-x   2 krush    csci        512 Jan 29 15:03 jan29
```

Display the number of lines where the pattern was found. This does not mean the number of occurrences of the pattern.

```
% ls -l | grep -c '^d'
10
```

Example: grep with \< \>

```
% cat grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
southern        SO      Suan Chin               54500.10
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print the line if it contains the word "north".

```
% grep '\<north\>' grep-datafile
north           NO      Ann Stephens            455000.50
```

Example: grep with a\|b

```
% cat grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
southern        SO      Suan Chin               54500.10
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print the lines that contain either the expression “NW” or the expression “EA”

```
% grep 'NW\|EA' grep-datafile
northwest       NW      Charles Main            300000.00
eastern         EA      TB Savage               440500.45
```

Note: egrep works with |

Example: egrep with +

```
% cat grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
southern        SO      Suan Chin               54500.10
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines containing one or more 3's.

```
% egrep '3+' grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
```

Note: grep works with \+

Example: egrep with RE: ?

```
% cat grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
southern        SO      Suan Chin               54500.10
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines containing a 2, followed by zero or one period, followed by a number.

```
% egrep '2\.?[0-9]' grep-datafile
southwest       SW      Lewis Dalsass           290000.73
```

Note: grep works with \?

Example: egrep with ( )

```
% cat grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
southern        SO      Suan Chin               54500.10
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines containing one or more consecutive occurrences of the pattern “no”.

```
% egrep '(no)+' grep-datafile
northwest       NW      Charles Main            300000.00
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
```

Note: grep works with \(\) \+

Example: egrep with (a|b)

```
% cat grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
southern        SO      Suan Chin               54500.10
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines containing the uppercase letter “S”, followed by either “h” or “u”.

```
% egrep 'S(h|u)' grep-datafile
western         WE      Sharon Gray             53000.89
southern        SO      Suan Chin               54500.10
```

Note: grep works with \(\) \|

Example: fgrep

```
% cat grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
southern        SO      Suan Chin               54500.10
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Find all lines in the file containing the literal string "[A-Z]****[0-9]..$5.00". All characters are treated as themselves. There are no special characters.

```
% fgrep '[A-Z]****[0-9]..$5.00' grep-datafile
Extra [A-Z]****[0-9]..$5.00
```

Example: Grep with ^

```
% cat grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
southern        SO      Suan Chin               54500.10
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines beginning with the letter n.

```
% grep '^n' grep-datafile
northwest       NW      Charles Main            300000.00
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
```

Example: grep with $

```
% cat grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
southern        SO      Suan Chin               54500.10
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines ending with a period and exactly two zero numbers.

```
% grep '\.00$' grep-datafile
northwest       NW      Charles Main            300000.00
southeast       SE      Patricia Hemenway       400000.00
Extra [A-Z]****[0-9]..$5.00
```

Example: grep with \char

```
% cat grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
southern        SO      Suan Chin               54500.10
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines containing the number 5, followed by a literal period and any single character.

```
% grep '5\..' grep-datafile
Extra [A-Z]****[0-9]..$5.00
```

Example: grep with [ ]

```
% cat grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
southern        SO      Suan Chin               54500.10
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines beginning with either a “w” or an “e”.

```
% grep '^[we]' grep-datafile
western         WE      Sharon Gray             53000.89
eastern         EA      TB Savage               440500.45
```

Example: grep with [^]

```
% cat grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
southern        SO      Suan Chin               54500.10
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines ending with a period and exactly two non-zero numbers.

```
% grep '\.[^0][^0]$' grep-datafile
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
eastern         EA      TB Savage               440500.45
```

Example: grep with x\{m\}

```
% cat grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
southern        SO      Suan Chin               54500.10
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines where there are at least six consecutive numbers followed by a period.

```
% grep '[0-9]\{6\}\.' grep-datafile
northwest       NW      Charles Main            300000.00
southwest       SW      Lewis Dalsass           290000.73
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
```

Example: grep with \<

```
% cat grep-datafile
northwest       NW      Charles Main            300000.00
western         WE      Sharon Gray             53000.89
southwest       SW      Lewis Dalsass           290000.73
southern        SO      Suan Chin               54500.10
southeast       SE      Patricia Hemenway       400000.00
eastern         EA      TB Savage               440500.45
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
central         CT      KRush                   575500.70
Extra [A-Z]****[0-9]..$5.00
```

Print all lines containing a word starting with “north”.

```
% grep '\<north' grep-datafile
northwest       NW      Charles Main            300000.00
northeast       NE      AM Main Jr.             57800.10
north           NO      Ann Stephens            455000.50
```

Example: egrep with linux.words

```
/usr/share/dict % egrep -i '^x.*x$' linux.words
Xerox
xerox
xix
xx
xxx
xylanthrax
```

Print all words that begin and end with ‘x’.

Example: egrep with linux.words

```
/usr/share/dict % egrep '.*sex.*' linux.words | wc
     325     325    3564
```

Counts all words that have “sex” as a substring

```
Some of the 325 words:
Essexville
misexample
misexecute
misexecution
misexpectation
misexpend
misexpenditure
misexplain
misexplained
misexplanation
```

Example: egrep with linux.words

```
/usr/share/dict % egrep '.*b.*b.*b.*b.*' linux.words
```

Lists words that have at least 4 b's in them

```
Some of the 25 words:
beerbibber
bibble-babble
blood-bedabbled
bubble-bow
bubblebow
bubbybush
bumblebomb
double-bubble
flibbertigibbet
flibbertigibbets
flibbertigibbety
gibble-gabble
gibblegabble
gibble-gabbler
gibblegabbler
hubble-bubble
```