CST8177 - Lab #5

Student Name:	**Student Number:**	**Lab section:**

Working with Regular Expressions (aka regex or RE)

<u>In-Lab Demo</u> - List all the non-user accounts in **`/etc/passwd`** that use **`/sbin`** as their home directory. State the purpose of each field in a password file entry - see passwd(5).

Overview

- Regular expressions are used for <u>pattern matching</u>.
- Regular expressions are interpreted by specific utilities, such as **`grep`**, and not by the shell. To prevent the shell from interpreting special characters, since some are the same ones the shell uses, use quotes when passing a regular expression as an argument.
  - *Examples*:
    - **`grep ro*t /etc/passwd`**
    - **`grep 'ro*t' /etc/passwd`**
- Regex metacharacters are different from file glob (wild card) metacharacters (although some, notably **`*`**, are the same character).
- **`grep`** stands for **g**lobal **r**egular **e**xpression and **p**rint, derived from the Unix text editor **`ed`** construct **`g/re/p`**.
- It will always match the FIRST and LONGEST string.

Summary of regexes of the <u>basic</u> set

	Meaning
`.`	Matches any single character (except newline, **`0x0A`**). *Example*: **`ro.t`** matches **`root`**, **`robt`**, **`ro3t`**, **`ro@t`**, and so on *Note*: The newline is not considered a printable character.
`*`	Matches zero or more of the <u>preceding</u> item (unlike in a file glob, it cannot stand alone; it always modifies the previous item) *Example:* the pattern **`ro*t`** matches **`rt`**, **`rot`**, **`root`**, **`rooot`** and so on for any number of **`o`** (but no other letter).
`[...]`	Matches any single character in the list (like file glob). *Example*: **`l[io]ve`** matches **`live`** or **`love`** but not **`lave`** or **`lrve`** *Note*: Ranges like **`a-z`** or **`0-9`** are valid as long as the start is lower in the ASCII list than the end (**`[0-2]`** is OK, **`[2-0]`** is not). Use **`LC_ALL=C`**. To use the range indicator **`-`** as a match character, escape it as **`\-`**.
`[^...]`	Matches any character **<u>not</u>** in the list. *Note*: If a caret (**`^`**) is in a **`[...]`** list but not at the beginning, it is interpreted as being just a normal character. It can also be escaped by **`\`**.
`\(...\)`	Group into an item. Used with **`|`**, select one item from a list
`\{n,m\}`	Match the preceding item at least '**`\{n\}`**' or more times; or exactly '**`\{n,\}`**' times; or using **`\{n,n\}`**, from **`n`** to **`m`** times.

^	Anchors the regex at the beginning of the line if the caret is the first regex character.
	Example: These will provide different output:
	`grep 'root' /etc/passwd`
	`grep '^root' /etc/passwd`

$	Anchors the regex at the end of the line if the dollar sign is the last regex character.
	Example: These will provide different output:
	`grep 'root' /etc/passwd`
	`grep 'root$' /etc/passwd`

'^$'	The regex to represent an empty line.

Exercise #1: Viewing regular expression output

Type the following 7 lines of text exactly in **vi** as the file **lab4-re** using the line-breaks given as **[Enter]** only (or copy/paste from the document, replacing **[ENTER]** and **[TAB]**, and ensuring that exactly 7 lines result):

```
How to Please your Technical Support Department[Enter]
Tip:[Enter]
When you call us to have your computer moved, leave it buried under
postcards and family pictures.[Enter]
We don't have a life and we are deeply moved when catching a glimpse of
yours.[Enter]
[Enter]
Thank you![Enter]
[Tab]Your IT Department (Call 555)[Enter]
```

Type the following commands (omit the comment - **#** and following), and record the line numbers 1 to 7 only, to observe the result of the commands. Note: The **-n** switch of **grep** displays the line number in addition to the line found, if any.

Example: `grep -n '^root:' /etc/passwd  # also try with another user id`

- `grep -n '.' lab4-re     # matches any line with any single char anywhere`

  ____

- `grep -n '\.' lab4-re    # matches any line with a (literal) period`

  ____

- `grep -n 'T' lab4-re     # matches any line with the character T`

  ____

- `grep -n '^T' lab4-re    # matches any line beginning with the char T`

  ____

- `grep -n '^[A-Z]...$' lab4-re # Match 4-letter line starting upper case`

  ____

- `grep -n '^[A-Z][a-z]*:' lab4-re   # Matches any alpha line with a colon`

  ____

- `grep -n '^$' lab4-re    # Matches any empty line`
- `grep -n '[Ii][Tt]' lab4-re    # Matches any line with IT, it, It, iT`
- `grep -n -i 'it' lab4-re       # Also matches as above`
- `grep -n '[0-9]' lab4-re       # matches any line containing a number`
- `grep -n 'call' lab4-re        # matches any line with the string`
- `grep -n 'ca.*l' lab4-re   # matches 0 or more char between 'ca' and 'l'`
- `grep -n 'cal*' lab4-re   # matches 'ca' followed by 0 or more 'l's`
- What is the difference between the last 2 regexes: They both use **c**, **a**, *****, and **l**?

Exercise #2: Searching a system file using grep

Use **grep** to search the password file for specific strings using regular expressions. As root, make a backup copy of your **/etc/passwd** file and create an account for each of the following users: **afoo**, **foo**, **foobar**. Read the information in **man 5 passwd** for details of the password file and its colon-separated fields, and **man 5 shadow** for the shadow password file. Hint: Anchor your regex on something solid, like the start or end of the line, or on the colon-separators, or both.

Record the regex and the output for each of the following actions:

- Display **root**'s account (only one line of output)
- Display **foo**'s account (only one line of output)
- Display **foobar**'s account (only one line of output)
- Display all accounts with **/sbin/nologin** as the shell (7th and last field) - list the userids

- Display all accounts with **/home** as the parent home directory (6th field) - list the userids

---

- _Search all accounts in the password or shadow file that have no valid password - list the userids; which file?

- Search all accounts in the password or shadow file that have a locked password - list the userids; which file?

Exercise #3: Extended REs

Some examples using the extended regular expression set: ORing

To work with the extended regular expression set, use **egrep** instead of **grep**. The pipe symbol is the regex OR operator and allows you to look for more than one pattern, in the form **(pattern-1|pattern-2|...|pattern-n)**. This OR is the inclusive or, and results in *true* if this or that or both are *true*. That is, if you evaluate **a | b** logically, when either **a** is *true* or **b** is *true* or both are *true*, the result is *true*.

Example: **egrep '^(root|bin):' /etc/passwd**

- Compare the example above with **egrep '(root|bin):' /etc/passwd**. If the results are different, why is this so?

- Display all accounts with group id of 100 or 500: **egrep "^[^:]*:[^:]*:[^:]*:(100|500):" /etc/passwd | cut -d : -f 1**

- Why or how does this regex work?

- Display all accounts with group id 0 to 100 (that is, a 1-digit number, or a 2-digit number, or a 3-digit number starting with the digit '**1**'):
  egrep "^[^:]*:[^:]*:[^:]*:([0-9]|[0-9][0-9]|100):[^:]*:[^:]*:[^:]*$" /etc/passwd | cut -d : -f 1

- Try this again with **egrep "^[^:]*:[^:]*:[^:]*:([0-9]|[0-9][0-9]|100):" /etc/passwd | cut -d : -f 1**

- Why or how does each regex work?

Working with some grep options

The **grep** utility has a number of options. Some of the most frequently used (there are lots more) include:

-c	displays a count of matching lines
-i	ignores the case or letters in making comparisons
-n	displays line number
-q	quiet: used when scripts collect the **exit** status **$?** as a POSIX alternative to redirecting output to **/dev/null**
-v	inverts the search to display only lines that do NOT match
-w	matches the string as a word

Experiment with the **grep** options above in addition to these samples.

grep -c "^" lab4-re and **grep -c "$" lab4-re**

How many lines are in the file **lab4-re**? Why or how do these regexes work?

What happens if you omit the regex and use **grep -c lab4-re**

grep -v "." lab4-re

Why or how does this regex work?

grep -v "\." lab4-re

Why or how does this regex work?

Using at least the **-v** option of grep, display only lines in **lab4-re** that do not contain the string "**you**". Show your **grep** command here:

Count all lines with the string "**you**" and separately, list only their line numbers. Show your two **grep** commands here (you may need to pipe **grep**'s output to another utility):

Did any of your "**you**" matches surprise you? Which and why?

(You may have to pretend to be easily surprised!)