OPS102 – Week 12 – Regular Expressions - Sample Lab

Student Name: Chetan Arora

Student ID: 100976240

Activity 1: Simple Regular Expressions

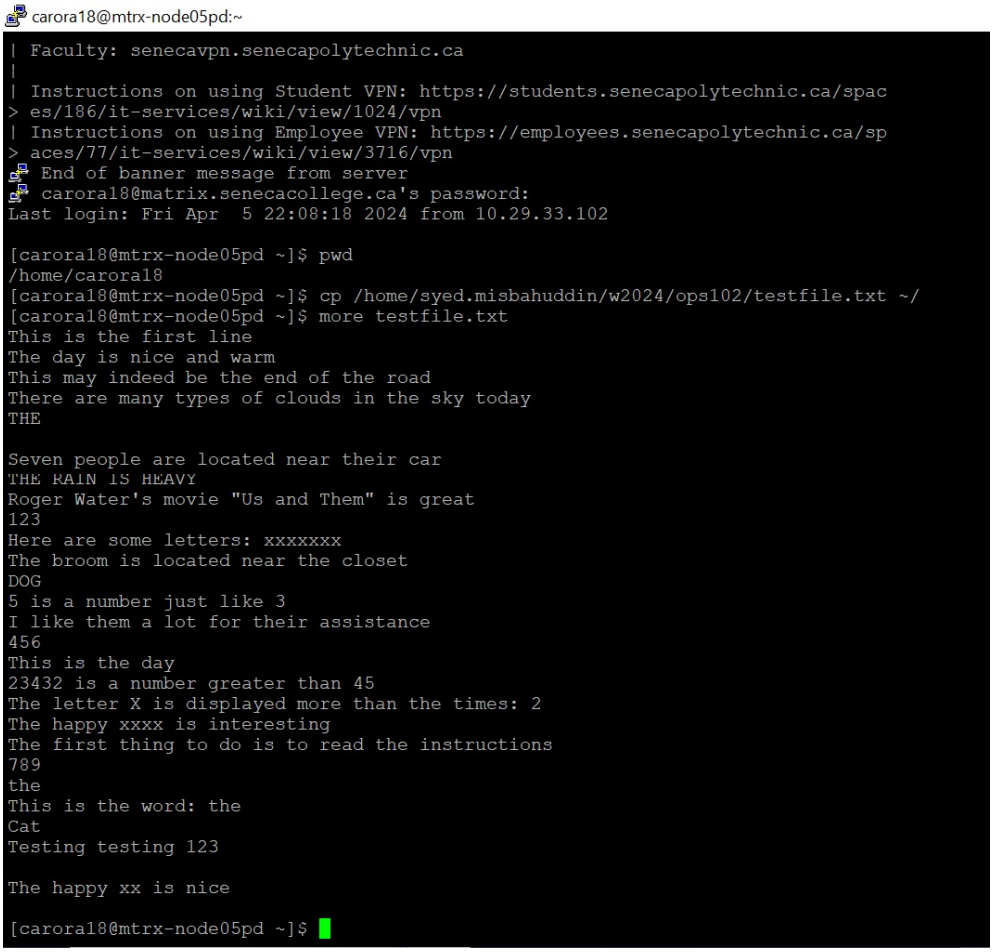
Perform the Following Steps:

1. Login to your matrix account.
2. Issue a Linux command to **confirm** you are located in your **home** directory.
3. Issue the following Linux command to **copy** a text file to *your* **home** directory from the OPS102 home directory:

   `cp ~ops102/datafiles/ testfile.txt  ~/`

   This text file contains some random text.

4. View the contents of this text file using more or less command.



```
This is the first line
The day is nice and warm
This may indeed be the end of the road
There are many types of clouds in the sky today
THE

Seven people are located near their car
THE RAIN IS HEAVY
Roger Water's movie "Us and Them" is great
123
Here are some letters: xxxxxxx
The broom is located near the closet
DOG
5 is a number just like 3
I like them a lot for their assistance
456
This is the day
23432 is a number greater than 45
The letter X is displayed more than the times: 2
The happy xxxx is interesting
The first thing to do is to read the instructions
789
the
This is the word: the
Cat
Testing testing 123

The happy xx is nice

testfile.txt (END)
```

5. Although there are several Linux commands that use regular expressions, we will be using the **egrep** command for this lab.
6. Issue the following Linux command to match the pattern **the** within **testfile.txt**: **`egrep 'the' testfile.txt`**. What output you observe? Observe the matched patterns. You will notice that the pattern "**the**" is matched including larger words like "**them**" and "**their**".

```
[carora18@mtrx-node01pd ~]$ egrep 'the' testfile.txt
This is the first line
This may indeed be the end of the road
There are many types of clouds in the sky today
Seven people are located near their car
The broom is located near the closet
I like them a lot for their assistance
This is the day
The letter X is displayed more than the times: 2
The first thing to do is to read the instructions
the
This is the word: the
[carora18@mtrx-node01pd ~]$
```

When I use the egrep command with the pattern 'the', it will match larger words like "them" and "their" because the pattern 'the' is found within those words

7. Next issue the egrep Linux command with the **-i** option to ignore case sensitively:
   egrep -i 'the' testfile.txt

   What do you notice is different when issuing this command?

```
carora18@mtrx-node01pd:~
The first thing to do is to read the instructions
the
This is the word: the
[carora18@mtrx-node01pd ~]$ egrep -i 'the testfile.txt
> ^C
[carora18@mtrx-node01pd ~]$ egrep -i 'the' testfile.txt
This is the first line
The day is nice and warm
This may indeed be the end of the road
There are many types of clouds in the sky today
THE
Seven people are located near their car
THE RAIN IS HEAVY
Roger Water's movie "Us and Them" is great
The broom is located near the closet
I like them a lot for their assistance
This is the day
The letter X is displayed more than the times: 2
The happy xxxx is interesting
The first thing to do is to read the instructions
the
This is the word: the
The happy xx is nice
[carora18@mtrx-node01pd ~]$
```

   When I issue the egrep command with the -i option to ignore case sensitivity, the difference lies in the fact that the matching process becomes case-insensitive. This means that the pattern "the" will match not only "the" but also "The", "tHe", "THE", etc., regardless of the case of the letters within the text file.

8. Next issue the **grep** command with the **-w** option to only match the pattern as a **word**, and not as a part of any larger word like “them” or “their”.
   Take the screenshot of output and paste here.

```
[carora18@mtrx-node01pd ~]$ egrep -w 'the' testfile.txt
This is the first line
This may indeed be the end of the road
There are many types of clouds in the sky today
The broom is located near the closet
This is the day
The letter X is displayed more than the times: 2
The first thing to do is to read the instructions
the
This is the word: the
[carora18@mtrx-node01pd ~]$
```

Activity 2: matching the pattern at the beginning and end of line.

You learnt about the anchors which can be used to force a match at the beginning or end of the line.

1. Issue the following Linux command:
   `grep -w -i "^the" testfile.txt`

   The **^** matches the word "**the**" (both upper or lowercase) at the beginning of the line.

```
[carora18@mtrx-node01pd ~]$ grep -w -i "^the" testfile.txt
The day is nice and warm
THE
THE RAIN IS HEAVY
The broom is located near the closet
The letter X is displayed more than the times: 2
The happy xxxx is interesting
The first thing to do is to read the instructions
the
The happy xx is nice
[carora18@mtrx-node01pd ~]$
```

2. Issue the following Linux command:
   `grep -w -i "the$" testfile.txt`

   The **$** symbol is used to anchor patterns at the end of the line.

```
[carora18@mtrx-node01pd ~]$ grep -w -i "the$" testfile.txt
THE
the
This is the word: the
[carora18@mtrx-node01pd ~]$
```

3. Issue the following Linux command to match the word "**the**" **simultaneously** at the beginning and end of the line:
   `grep -w -i "^the$" testfile.txt`

```
[carora18@mtrx-node01pd ~]$ grep -w -i "^the$" testfile.txt
THE
the
[carora18@mtrx-node01pd ~]$
```

   What did you notice? Did you receive any output? This essentially means that we are searching for lines in a file that exactly match the word "the". Let's break down the command:

   - '^the$': This is the pattern or regular expression that is being searched for. In this case, it consists of two parts:

   - ^ is a special character in regular expressions that denotes the start of a line.
   - 'the' is the literal word being searched for.
   - $ is another special character in regular expressions that denotes the end of a line.
   - So, the complete pattern '^the$' ensures that only lines containing the exact word "the" (with no other characters before or after) will be matched.

4. Next issue a command to match all the empty lines in the file hp.txt. What command it should be. Take a screenshot of your command and paste below.

5. Issue the following Linux command to match strings that **begin and end with 3 characters**:
   `grep "^...$" testfile.txt`
   What did you notice?

```
[carora18@mtrx-node01pd ~]$ grep "^...$" testfile.txt
THE
123
DOG
456
789
the
Cat
[carora18@mtrx-node01pd ~]$
```

   these lines contain exactly three characters each.

Activity 3: Using atoms and wild cards to match approximate patterns

Suppose you were interested in discovering three-letter words that begin with ‘c’ and end with ‘t’. This illustrates a type of matching that is not literal or stringent, unlike our previous activity where we specifically sought out the exact word "the."

1. Run the following egrep command:
   `egrep 'c.t'-i testfile.txt`
   In this example we used the dot ‘.’which matches any single character except a newline.
   What output do you observe, which words are matched as a result of this command?
   Answer below.

   Run the same egrep command with “-i” and “-w” options.

```
[carora18@mtrx-node01pd ~]$ egrep -i "c.t" testfile.txt
Seven people are located near their car
The broom is located near the closet
Cat
[carora18@mtrx-node01pd ~]$
```

   In this command, the pattern 'c.t' is specified, where the dot (.) matches any single character except a newline, and the -i option makes the search case-insensitive.

```
carora18@mtrx-node01pd ~]$ egrep "c.t" -w testfile.txt
carora18@mtrx-node01pd ~]$ egrep -w 'c.t' testfile.txt
carora18@mtrx-node01pd ~]$ egrep "c.t" -w testfile.txt
carora18@mtrx-node01pd ~]$ egrep -w 'c.t' testfile.txt
carora18@mtrx-node01pd ~]$
```

2. Let’s to match even more relaxed pattern; all three letter words which begin with the character ‘a’ .
   Run the following egrep command.
   `egrep 'a..'-i  testfile.txt`

With this command you will see many more matches are found, try to reduce the matches by using "-w" option to find the three letter words which are not part of a bigger word.

```
[carora18@mtrx-node01pd ~]$ egrep "a.." -i  testfile.txt
The day is nice and warm
This may indeed be the end of the road
There are many types of clouds in the sky today
Seven people are located near their car
THE RAIN IS HEAVY
Roger Water's movie "Us and Them" is great
Here are some letters: xxxxxxx
The broom is located near the closet
5 is a number just like 3
I like them a lot for their assistance
23432 is a number greater than 45
The letter X is displayed more than the times: 2
The happy xxxx is interesting
The first thing to do is to read the instructions
The happy xx is nice
[carora18@mtrx-node01pd ~]$
```

```
[carora18@mtrx-node01pd ~]$ egrep "a.." -w  testfile.txt
The day is nice and warm
There are many types of clouds in the sky today
Seven people are located near their car
Roger Water's movie "Us and Them" is great
Here are some letters: xxxxxxx
[carora18@mtrx-node01pd ~]$
```

3. Let's issue a Linux command to display strings that contain **one or more consecutive occurrence** of the letter "**t**":

   `egrep "tt*" testfile.txt`

   Why did this work? because the pattern indicates one occurrence of the letter "t", followed by **zero or MORE occurrences** of the next letter "t".
   The wild card character asterisk is used to match zero or more occurrences of preceding pattern.

```
[carora18@mtrx-node01pd ~]$ egrep "tt*" testfile.txt
This is the first line
This may indeed be the end of the road
There are many types of clouds in the sky today
Seven people are located near their car
Roger Water's movie "Us and Them" is great
Here are some letters: xxxxxxx
The broom is located near the closet
5 is a number just like 3
I like them a lot for their assistance
This is the day
23432 is a number greater than 45
The letter X is displayed more than the times: 2
The happy xxxx is interesting
The first thing to do is to read the instructions
the
This is the word: the
Cat
Testing testing 123
[carora18@mtrx-node01pd ~]$
```

text, test, teeeest: Matches because they contain one or more consecutive occurrences of the letter "t".
t, tt, ttt: Matches because they contain one occurrence of the letter "t" followed by zero or more additional "t"s.

4. Another way to search for exact same pattern is to use the wild card character ‘a’, which matches one or more occurrences of preceding pattern.
   Run the following egrep command.
   `egrep "t+" testfile.txt`
   What output do you see? Explain your observation.

```
[carora18@mtrx-node01pd ~]$ egrep "t+" testfile.txt
This is the first line
This may indeed be the end of the road
There are many types of clouds in the sky today
Seven people are located near their car
Roger Water's movie "Us and Them" is great
Here are some letters: xxxxxxx
The broom is located near the closet
5 is a number just like 3
I like them a lot for their assistance
This is the day
23432 is a number greater than 45
The letter X is displayed more than the times: 2
The happy xxxx is interesting
The first thing to do is to read the instructions
the
This is the word: the
Cat
Testing testing 123
```

Activity 4: Using Character Class to Match Approximate Patterns

Suppose you were interested in discovering three-letter words that start with ‘c’, end with ‘t’ and have a vowel in the middle. There are five vowels, ‘a, e, I, o, u’. It could be any vowel in the three letter words. Character classes in Linux regular expressions allow you to specify a set of characters that can match a single character at that position.

1. Run the following Linux command.
   `egrep "c[aeious]t" testfile.txt`
   What output do you see? Explain your observation.

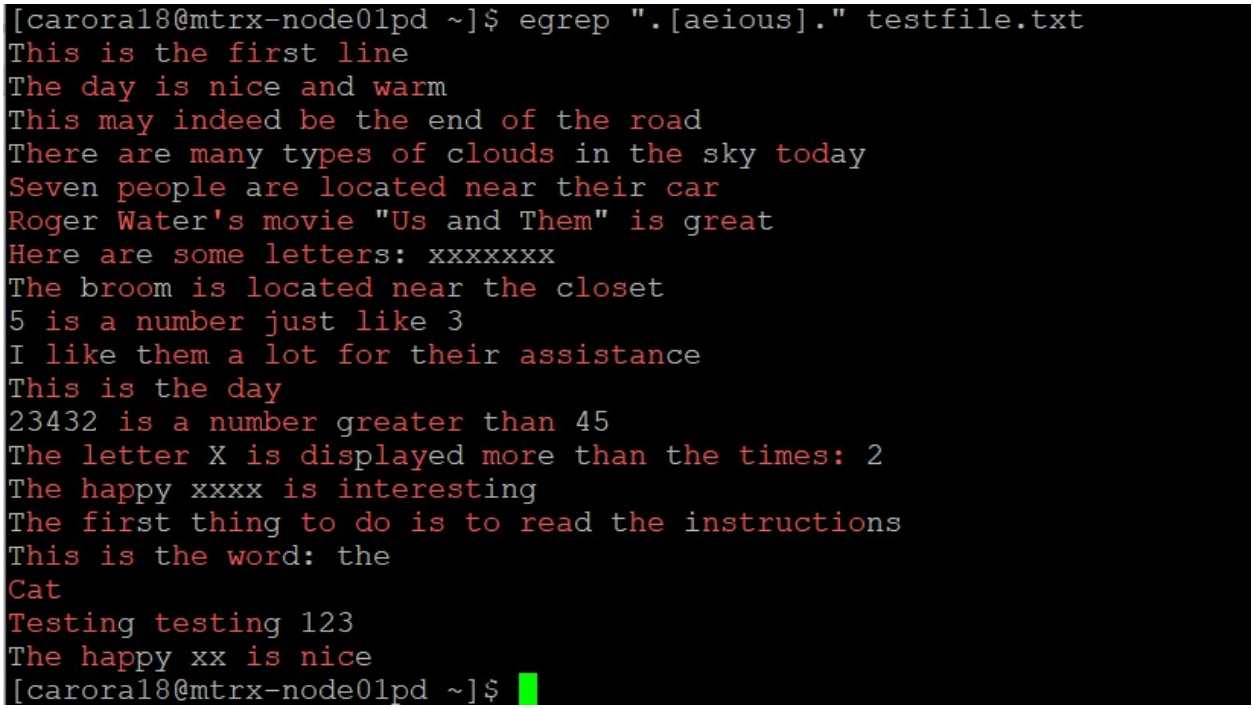
```
[carora18@mtrx-node01pd ~]$ egrep "c[aeious]t" testfile.txt
Seven people are located near their car
The broom is located near the closet
```

   The pattern "c[aeious]t" indicates that the character immediately after "c" can be any of the characters within the brackets [aeious], followed by "t". Each character within the brackets [aeious] is considered as a potential match.

2. Next issue a linux command to search for any three letter words that have a vowel in the middle. This is a more relaxed condition, we are no more looking for the three letter words which begin with a specific character like ‘c’ and end with a specific character like ‘t’. Use the dot character to match any character at the beginning and end.
   Run this command.

   `egrep ".[aeious]." testfile.txt`

   What output do you see? Explain your observation. Did you also find the matches like “am” or “ he”? This is because dot matches any character except for the new line character. In the matches like “am” and “he”, the third character is actually the space. Can you modify the regular expression to match the three letter words excluding the space as a valid character?
   What command it should be? Run your command, take its screenshot and paste below.



   This command searches for strings in testfile.txt that contain any character followed by a vowel followed by any character.

   The pattern ".[aeious]." consists of:

   “.” Matches any single character except for a newline character.
   [aeious]: Matches any single character within the brackets, which in this case are the vowels 'a', 'e', 'i', 'o', and 'u'.
   “.” Again, matches any single character except for a newline character.
   So, the command matches strings that contain any character followed by a vowel followed by any character.

3. Issue the following Linux command to match strings that **consist of only 3 digits**:
   egrep "^[0-9][0-9][0-9]$" testfile.txt

```
[carora18@mtrx-node01pd ~]$ egrep "^[0-9][0-9][0-9]$" testfile.txt
123
456
789
```

4. Issue the following Linux pipeline command to display **signed** or **unsigned integers**:
   egrep "^[+-][0-9]+$" testfile.txt

```
[carora18@mtrx-node01pd ~]$ egrep "^[+-][0-9]+$" testfile.txt
[carora18@mtrx-node01pd ~]$ egrep "^[+-][0-9] +$" testfile.txt
[carora18@mtrx-node01pd ~]$ egrep '^[+-][0-9]+$' testfile.txt
```

Activity 5: Using Repetition to Match Approximate Patterns

Curly braces {} - Match a specific number of occurrences of the preceding element. Here, the regular expression pattern **a{3}** specifies that "a" should appear exactly three times and the output should be:

aaa

Only the line containing exactly three "a" character is matched.

1. Issue the following Linux command to match exactly three consecutive upper case letters.
   egrep '[A-Z]{3}' testfile.txt

```
[carora18@mtrx-node01pd ~]$ egrep "[A-Z]{3}" testfile.txt
THE
THE RAIN IS HEAVY
DOG
```

2. To search for the lines that have exactly two digits, you can use the following command:
   egrep '[0-9]{2}' testfile.txt

```
[carora18@mtrx-node01pd ~]$ egrep "[0-9]{2}" testfile.txt
123
456
23432 is a number greater than 45
789
Testing testing 123
[carora18@mtrx-node01pd ~]$
```

3. Write a regular expression pattern that matches a valid phone number with an area code. The area code should consist of exactly 3 digits, followed by a hyphen ("-"), and then the phone number should consist of exactly 7 digits. Use curly braces for repetition. Take a screenshot of your command and its output and paste it below.
   egrep -i "^[0-9]{3}-[0-9]{7}$" testfile.txt

Activity 6: Using Alternations in regular expressions

Alternation is a key feature in regex that allows you to specify multiple alternative patterns using the pipe character "|". Alternation allows you to match one pattern or another, providing flexibility in defining complex search criteria.

The pipe character "|" acts as the alternation operator. It separates alternative patterns, indicating that the expression should match either the pattern on the left or the pattern on the right.

To match either "cat" or "dog", you can use the regex: cat|dog

Grouping Alternations: Parentheses can be used to group alternations, allowing more complex combinations of patterns.

Example: To match either "apple pie" or "raspberry pie", you can use the regex: (apple|raspberry)pie

1. Issue the following Linux command to match either a date in "mm/dd/yyyy" format or "dd-mm-yyyy" format:

   egrep '[0-9]{2}/[0-9]{2}/[0-9]{4}|[0-9]{2}-[0-9]{2}-[0-9]{4}' testfile.txt

   What do you observe?

   ```
   [caroral8@mtrx-node01pd ~]$ egrep "[0-9]{2}\/[0-9]{2}\/[0-9]{4}|[0-9]{2}-[0-9]{2}-[0-9]{4}" testfile.txt
   [caroral8@mtrx-node01pd ~]$ egrep "[0-9]{2}\/[0-9]{2}\/[0-9]{4}|[0-9]{2}-[0-9]{2}-[0-9]{4}" testfile.txt
   [caroral8@mtrx-node01pd ~]$
   ```

   Not working

2. What do the following regular expression match?

   'gr(a|e)y'
   'I love apples|bananas'
   'I love(apples|bananas)'

```
[caroral8@mtrx-node01pd ~]$ egrep 'gr(a|e)y' testfile.txt
[caroral8@mtrx-node01pd ~]$ egrep "gr(a|e)y" testfile.txt
[caroral8@mtrx-node01pd ~]$
```

Not working