CMPS 3500 Programming Languages
Activity 3
Task 5

1. Explain, with several examples, how regex.perl uses regex to accomplish its purpose.

Regex.perl reads lines from a file called nyse.in, processes the text using regular expressions, and writes the modified text to another file called nyse.out. It cleans up the text by removing extra spaces and special characters as well as rearrange and format the content of each line by reordering words and adding quotes. It also identifies and prints specific patterns. And here are some examples:

a) **It removes extra spaces by using this line: $line =~ s/ +/ /g; So if we had: ACN Accenture Ltd. before, then after it would be: ACN Accenture Ltd.**

b) **It rearranges substrings and added quotes using this line: $line =~ s/(^[0-9]*) ([A-Z]{3}) (.*$)/$2 $1 "$3"/; First it looks for a line that starts with numbers (^[0-9]*), followed by a space, three capital letters ([A-Z]{3}), and then the rest of the line (.*$). Then it swaps the first two parts of the line and puts quotes around the third part. So if we had: 7776 ACN Accenture Ltd. before, then it would be: ACN 7776 "Accenture Ltd." after.**

c) **It removes everything after a comma, including it with this line: $line =~ s/,.*$/"/; Here it finds anything after a comma (,.*) and removes it, but adds a closing quotation mark ("). For example, if we had: 9699 ANF Abercrombie&Fitch, S.A. before, then it will be changed to ANF 9699 "Abercrombie and Fitch" - there are actually more changes happening in this line, described in other examples.**

d) **It adds spaces between lowercase and uppercase letters in this line: $line =~ s/([a-z])([A-Z])/$1 $2/; It ensures that collapsed words are separated properly. For example if we had: AccentureLtd before, it will be: Accenture Ltd after.**

e) **It removes special characters like &, - and " with these line: $line =~ s/&/ and /; $line =~ s/-/ /;$line =~ s/'/ /; and replaces & with and, - with a space and ' is completely removed.**

f) **It removes periods with this line: $line =~ s/\./ /g; and replaces it with a space.**

g) It translates DOS linefeeds (carriage returns) into a null character with this line: $line =~ tr/\15/\00/; to ensure that line endings are handled correctly.

h) It extracts fields and matching patters with these lines: if ($line =~ /7[1-5]/){ print "field1: $field1 field2: $field2 \n"; by checking if the line contains the pattern 7[1-5], which means it's looking for any number that starts with 7 and is followed by a digit between 1 and 5. If the pattern is found, it prints the first two fields from the line.

2. Provide a list of the regular expressions, with some examples, that you used to clean up PhoneBook.txt.

a) I used $currentLine =~ tr/\15/\00/; - to remove control characters. This regex translates or replaces the carriage return character (\15, common in DOS-based systems) with a null character (\00), which is removing it from the line. If before was: "Kimberlee Turlington,039 298-72-30\r" The after would be : "Kimberlee Turlington,039 298-72-30"

b) Using $nameField =~ s/,//g; I was able to remove commas in the name field, making sure the names don't contain any punctuation. If before was: "Kimberlee Turlington," then after would be: "Kimberlee Turlington"

c) I used $nameField =~ s/[^a-zA-Z.\s\-]//g; to remove characters from the name that are not letters (a-zA-Z), spaces (\s), periods (.), or hyphens (-). It makes sure that the names contain only letters and valid punctuation. So if before I had: "Kimberlee Turlington123", then after would be: "Kimberlee Turlington"

d) Using ($lastName, $firstName) = split " ", $nameField, 2; I was able to split the name string by spaces and assigns the first word to $lastName and the second word to $firstName, ignoring any extra words. So if before I had: "Kimberlee Turlington Evelyn", then afterward it would be: $lastName = "Kimberlee", $firstName = "Turlington"

e) I used $rawPhoneNumber =~ s/[^0-9]//g; to remove all characters from the phone number that are not digits, leaving only the numbers. Therefore if before I had: "039 298-72-30", then after it would be: 0392987230

f) Using $rawPhoneNumber = sprintf '%s%s', "661", $rawPhoneNumber; I was able to add 661 to 7 digits numbers, and this line $rawPhoneNumber = substr $rawPhoneNumber, 1; to drop first digit from 11-digit numbers based on a few if-else checks.

g) Using these line: if ($leadingDigit == "0") {
$rawPhoneNumber = substr $rawPhoneNumber, 1;
$rawPhoneNumber = sprintf '%s%s', "92", $rawPhoneNumber;

} for the 9-digit numbers that start with a "0", I was able to remove the leading "0" and replace it with 92 as a new prefix. For example, if I had 039298723 before, then it would become 9239298723 after.

3. How many phone numbers where dropped?

According to my code 3635 phone numbers were dropped.

4. How many phone numbers where correctly identified?

According to my code 5953 phone numbers were correctly identified.

5. How many phone numbers of 10 digits where correctly identified?

According to my code 2349 of 10 digit phone numbers were correctly identified.