

PART 1:

Question 1:

Here is a screenshot of my regular expression and it matching all of the test cases. It's not hard coded, it accepts all strings that fit the pattern and rejects all those that don't. My regex in non screenshot form, `/^DT NN (WP VBD DT NN *)+$/`

The screenshot shows a terminal window in VS Code running on WSL (Debian). The terminal title is "hw3 [WSL: Debian]". The user has run the command `perl question_one.pl DT NN WP VBD DT NN` and is seeing the output of the script. The terminal interface includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is active), and PORTS.

```
home > rthompson > arizona > ht538-computational_linguistics > hw4 > question_one.pl
1 $s = qq/@ARGV/; print "Yes\n" if $s =~ /^DT NN (WP VBD DT NN *)$/;
2

rthompson@DESKTOP-EAHC8Q8:~/arizona/ht538-computational_linguistics$ perl question_one.pl DT NN WP VBD DT NN
Yes
rthompson@DESKTOP-EAHC8Q8:~/arizona/ht538-computational_linguistics$ perl question_one.pl DT NN WP VBD DT NN WP VBD DT NN
Yes
rthompson@DESKTOP-EAHC8Q8:~/arizona/ht538-computational_linguistics$ perl question_one.pl DT NN WP VBD DT NN WP VBD DT NN WP VBD DT NN
Yes
rthompson@DESKTOP-EAHC8Q8:~/arizona/ht538-computational_linguistics$ perl question_one.pl DT DT NN WP VBD DT NN
rthompson@DESKTOP-EAHC8Q8:~/arizona/ht538-computational_linguistics$ perl question_one.pl DT NN WP VBD DT NN NN
rthompson@DESKTOP-EAHC8Q8:~/arizona/ht538-computational_linguistics$ perl question_one.pl JJ DT NN
rthompson@DESKTOP-EAHC8Q8:~/arizona/ht538-computational_linguistics$ perl question_one.pl DT VBD NN
rthompson@DESKTOP-EAHC8Q8:~/arizona/ht538-computational_linguistics$ perl question_one.pl DT NN DT WP VBD DT NN
rthompson@DESKTOP-EAHC8Q8:~/arizona/ht538-computational_linguistics$
```

Question 2:

The screenshot shows a terminal window with the following content:

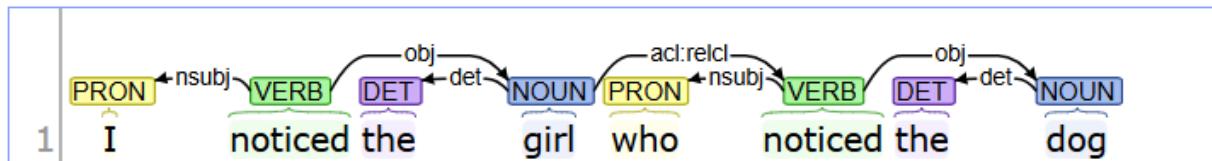
```
File Edit Selection View ... ← → 🔍 hw3 [WSL:Debian] 🗃 🔍 question_one.pl 🗃 question_2.pl ×  
home > rthompson > arizona > hlt538-computational_linguistics > hw4 > question_2.pl  
1 my $s = qq(@ARGV);  
2  
3 if ($s =~ /^DT NN (?<rec>DT NN(?: (?&rec))?) VBD$/) {  
4     print "$1\n";  
5 }  
  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ②  
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_2.pl DT NN DT NN DT NN VBD VBD  
1  
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_2.pl DT NN DT NN VBD  
1  
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_2.pl DT NN DT NN DT NN VBD VBD VBD VBD  
1  
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_2.pl DT DT NN DT NN VBD JJ  
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_2.pl DT NN DT NN VBD VBD  
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_2.pl DT NN VBD  
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_2.pl DT NN VBD VBD  
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_2.pl DT NN DT NN VBD VBD VBD VBD  
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_2.pl DT NN DT NN VBD DT NN DT NN VBD  
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$
```

Here is my regex “`^DT NN (?<rec>DT NN(?: (?&rec))?)? VBD$`” accepting and rejecting all of the corresponding test cases. By the Chomsky hierarchy, regular expressions shouldn’t be able to define this as it’s infix, but Perl REGEX aren’t exactly pure regular expressions. I spend way too much time googling capture groups to make this work and it took up the bulk of time for this assignment.

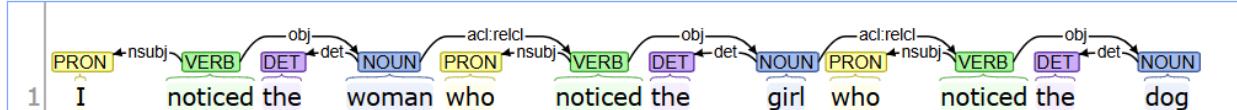
Question 3: The parser I used was <https://stanza.stanford.edu/>

3a, 3b, and 3c:

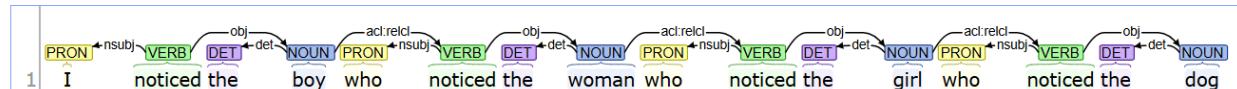
Universal Dependencies:



Universal Dependencies:



Universal Dependencies:

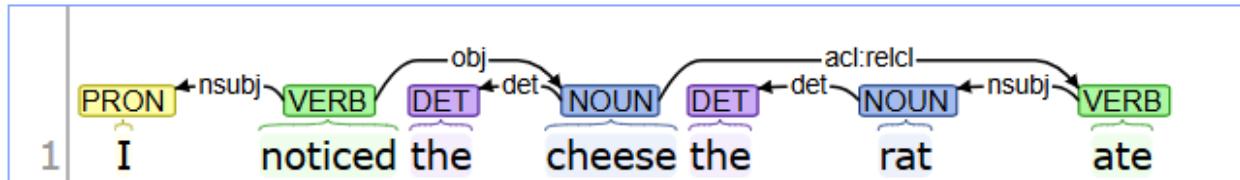


It does seem to parse all of them correctly. The chain of direct objects is unbroken all the way to the subject of the sentence. It’s a repeated pattern that doesn’t have to arc back to the original, which seems to make it easy for the parser.

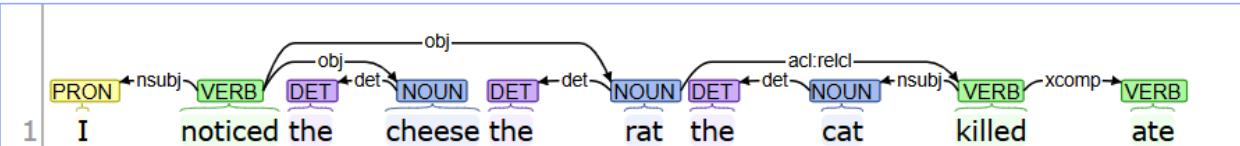
Question 4:

Here are 3a, 3b, and 3c for question 2:

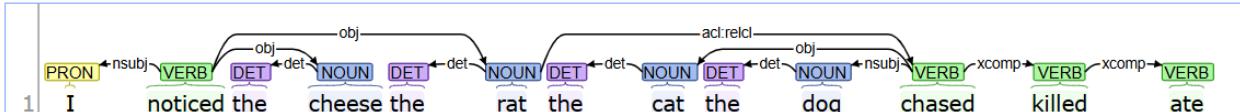
Universal Dependencies:



Universal Dependencies:



Universal Dependencies:



We appear to have broken it even starting at the second level. The verb ate is not connected to the noun cheese at all, even though that's what it should be the direct object of. It makes sense this is harder for the online parser to put together, because it's also harder for my internal parser to make sense of.

PART 2

Question 5:

The screenshot shows a terminal window in VS Code with the following content:

```
home > rthompson > arizona > hlt538-computational_linguistics > hw4 > question_5.pl
```

```
1 $n = shift;
2 $u = "1" x $n;
3 if ($u =~ /^(11+?)\1+$/) {
4     $f = length($1);
5     print "$n is divisible by $f\n";
6 } else {
7     print "$n is prime\n";
8 }
```

Below the code, the terminal shows the execution of the script with two different inputs:

- For input 256, the output is: 256 is divisible by 2
- For input 25600000, the output is: 25600000 is divisible by 400

The terminal tab bar includes PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS.

Here's the code. Using the length of the \$1 scalar yields the factor pretty easily.

Question 6:

The screenshot shows a terminal window with several tabs at the top: question_one.pl, question_2.pl, question_5.pl, question_6.pl (selected), and question_6.pl X. The current working directory is home > rthompson > arizona > hlt538-computational_linguistics > hw4 > question_6.pl. The code in question_6.pl is:

```
1  #!/usr/bin/perl
2  use strict;
3  use warnings;
4
5  my $count = 0;
6  while (<>) {
7      while ($_ =~ /\w+\/(\w*\|)*\b(VB|VBD|VBG|VBN|VBP|VBZ)\b\w*/g) {
8          $count++;
9      }
10 }
11 print "Total verbs: $count\n";
12
13
```

Below the code, there are navigation buttons: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (selected), and PORTS 2. The terminal history shows:

- rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4\$ perl question_6.pl HW4_wsj_pos.txt/ws_j_pos.txt
Total verbs: 66272
- rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4\$ █

Here's a screenshot of my verb counter. Total verbs is 66272. Hope that's the right amount.
Regex in text is `/\w+\/(\w*\|)*\b(VB|VBD|VBG|VBN|VBP|VBZ)\b\w*/`

Question 7:

The screenshot shows a Visual Studio Code (VS Code) interface running on a WSL (Windows Subsystem for Linux) environment. The title bar indicates the window is titled "hw3 [WSL: Debian]". The Explorer sidebar on the left shows a folder named "HW3 [WSL: DEBIAN]" containing files: "hw3-template-palindrome.py", "hw3.pdf", and "palindrome.pl". The main editor area displays a Perl script named "question_7.pl". The script reads from a file "HMM_wsj_pos.txt" and prints the top 20 verbs along with their counts. The terminal below the editor shows the output of the script:

```
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_7.pl HMM_wsj_pos.txt/wsj_pos.txt
says    1872
had     941
were   964
been   852
s      593
do     479
say    481
make   317
did    310
made   383
rose   278
does   277
expected 256
```

I did this one a little differently than in the video but this made more sense to me. I'm curious to see if the counts are right and if not, whether the problem is in the REGEX itself or in the code around it.