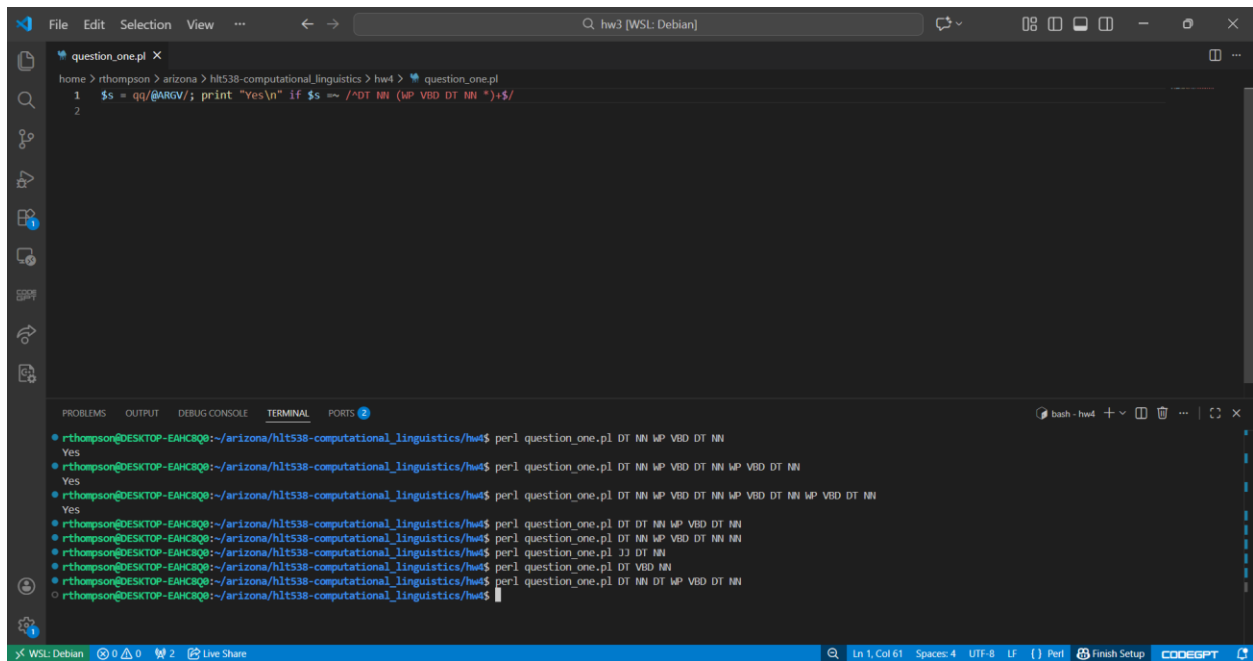


PART 1:

Question 1:

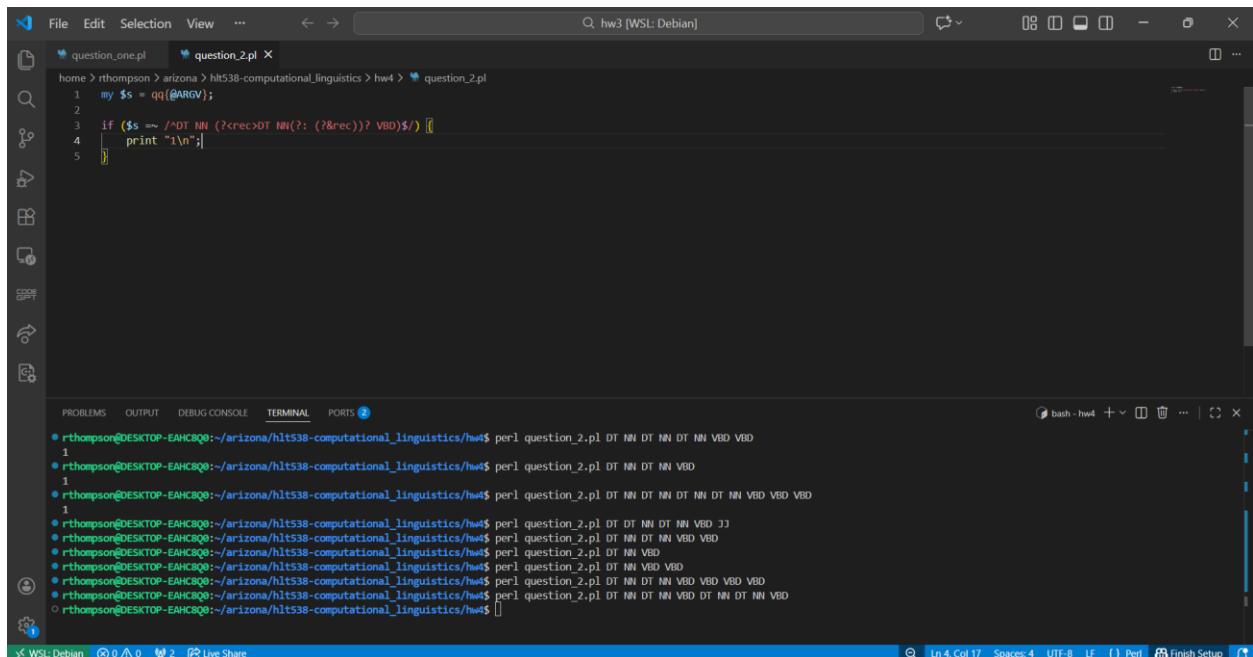
Here is a screenshot of my regular expression and it matching all of the test cases. It's not hard coded, it accepts all strings that fit the pattern and rejects all those that don't. My regex in non screenshot form, `/^DT NN (WP VBD DT NN *)+$/`



```
File Edit Selection View ... hw3 [WSL: Debian]
question_one.pl X
home > rthompson > arizona > hlt538-computational_linguistics > hw4 > question_one.pl
1 $s = qq{@ARGV}; print "Yes\n" if $s =~ /^DT NN (WP VBD DT NN *)+$/;
2

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
• rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_one.pl DT NN WP VBD DT NN
Yes
• rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_one.pl DT NN WP VBD DT NN WP VBD DT NN
Yes
• rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_one.pl DT NN WP VBD DT NN WP VBD DT NN WP VBD DT NN
Yes
• rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_one.pl DT DT NN WP VBD DT NN
• rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_one.pl DT NN WP VBD DT NN NN
• rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_one.pl JJ DT NN
• rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_one.pl DT VBD NN
• rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_one.pl DT NN DT WP VBD DT NN
• rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$
```

Question 2:



```
File Edit Selection View ... hw3 [WSL: Debian]
question_one.pl question_2.pl X
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 }

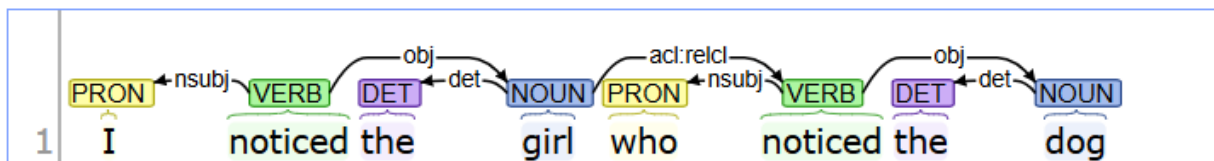
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
• rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_2.pl DT NN DT NN DT NN VBD VBD
1
• rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_2.pl DT NN DT NN VBD
1
• rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_2.pl DT NN DT NN DT NN VBD VBD VBD
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 infixed, 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.

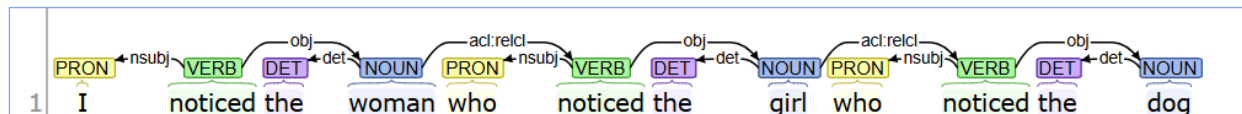
Question 3: The parser I used was <https://stanza.stanford.edu/>

3a, 3b, and 3c:

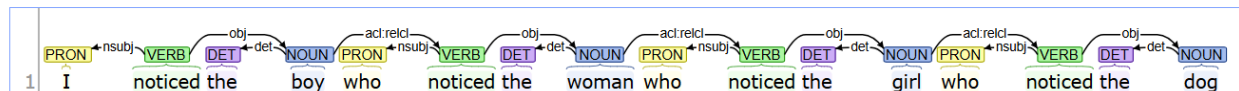
Universal Dependencies:



Universal Dependencies:



Universal Dependencies:

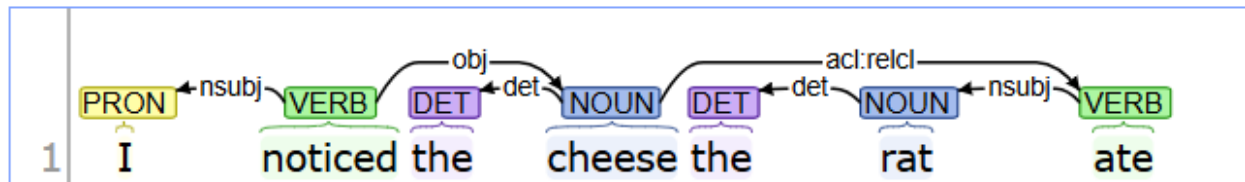


It does seem to parse all of them correctly. The chain of direct objects is unbroken all the way to the subject of the sentence. It’s a repeated pattern that doesn’t have to arc back to the original, which seems to make it easy for the parser.

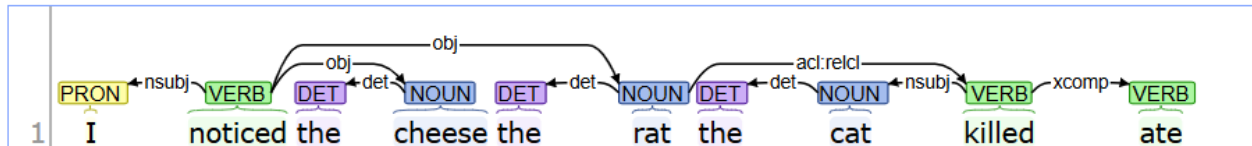
Question 4:

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

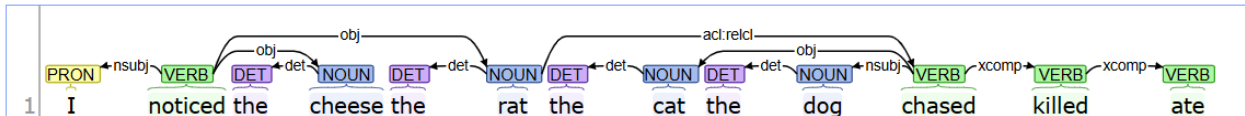
Universal Dependencies:



Universal Dependencies:



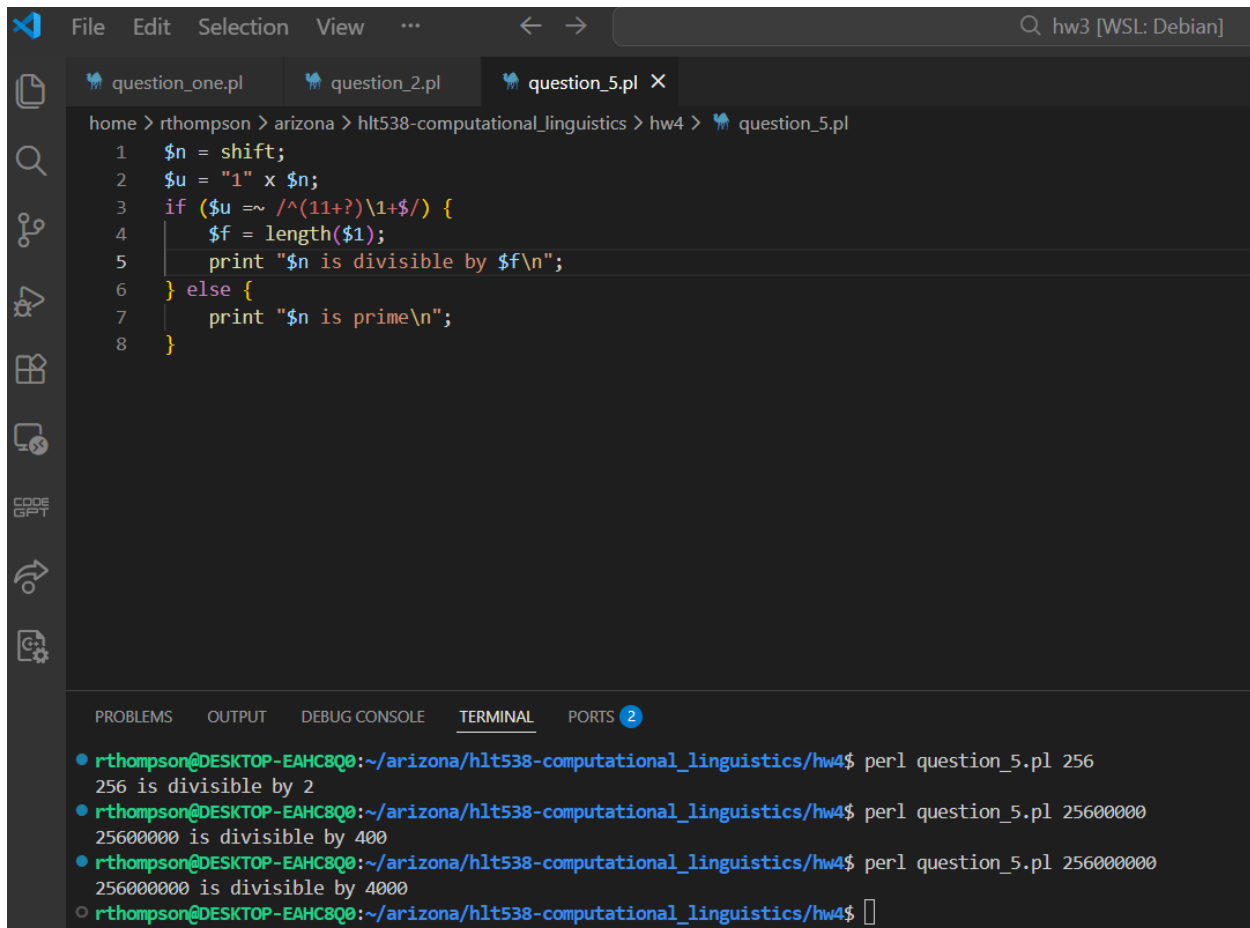
Universal Dependencies:



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 Visual Studio Code editor window with a dark theme. The top menu bar includes File, Edit, Selection, View, and a search bar containing 'hw3 [WSL: Debian]'. The Explorer sidebar on the left shows three files: question_one.pl, question_2.pl, and question_5.pl. The main editor area displays the content of question_5.pl, which is a Perl script. The script starts with a 'shift' command, then assigns the value of the first argument to a scalar variable \$u. It then uses a regular expression to check if the number is divisible by 2. If it is, it calculates the factor and prints a message. Otherwise, it prints that the number is prime. The bottom panel shows the TERMINAL output, which displays the results of running the script with three different inputs: 256, 25600000, and 256000000. The output shows that 256 is divisible by 2, 25600000 is divisible by 400, and 256000000 is divisible by 4000.

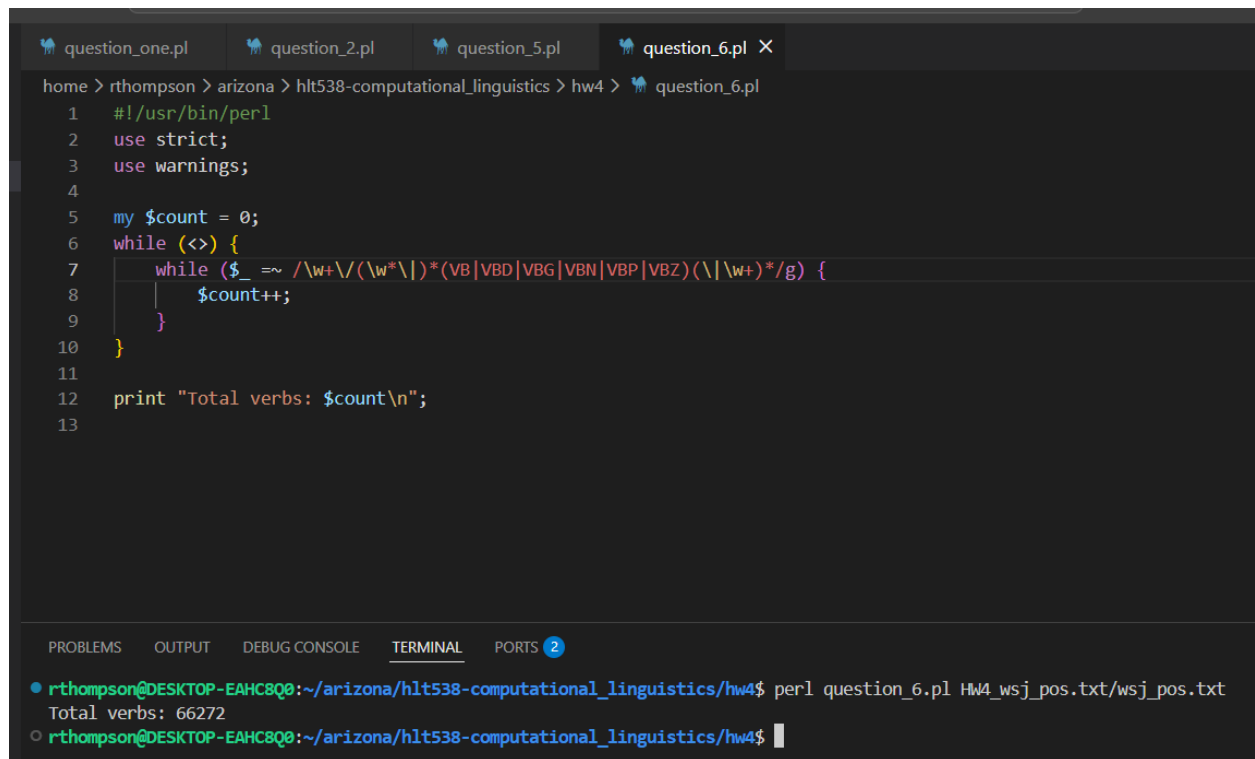
```
home > rthompson > arizona > hlt538-computational_linguistics > hw4 > question_5.pl
1  $n = shift;
2  $u = "1" x $n;
3  if ($u =~ /^(11+?)\1+$/) {
4      $f = length($1);
5      print "$n is divisible by $f\n";
6  } else {
7      print "$n is prime\n";
8  }
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS 2

- rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4\$ perl question_5.pl 256
256 is divisible by 2
- rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4\$ perl question_5.pl 25600000
25600000 is divisible by 400
- rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4\$ perl question_5.pl 256000000
256000000 is divisible by 4000
- rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4\$

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

Question 6:



The screenshot shows a code editor with four tabs: question_one.pl, question_2.pl, question_5.pl, and question_6.pl. The active tab is question_6.pl, which contains a Perl script. The script starts with a shebang line, uses strict and warnings, initializes a counter, and then enters a while loop that reads from a file and counts verbs using a regex. The output of the script is shown in the terminal panel at the bottom, which displays the total number of verbs as 66272.

```
home > rthompson > arizona > hlt538-computational_linguistics > hw4 > question_6.pl
1  #!/usr/bin/perl
2  use strict;
3  use warnings;
4
5  my $count = 0;
6  while (<>) {
7      while ($_ =~ /\w+\/(\w*\|)* (VB|VBD|VBG|VBN|VBP|VBZ)(\| \w+)* /g) {
8          $count++;
9      }
10 }
11
12 print "Total verbs: $count\n";
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 2

- rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4\$ perl question_6.pl Hw4_ws_j_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+V(\w*\|)* (VB|VBD|VBG|VBN|VBP|VBZ)(\| \w+)* /`

Question 7:

The screenshot shows a VS Code editor window with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'HW3 [WSL: DEBIAN]' with files 'hw3-template-palindrome.py', 'hw3.pdf', and 'palindrome.pl'. The main editor displays a Perl script named 'question_7.pl' with the following code:

```
1 #!/usr/bin/perl
2 use strict;
3 use warnings;
4
5 my %verbs;
6 my $count = 0;
7 while (<>) {
8     while ($ _ =~ /\b(w+)\b(?!(?:VB|VBD|VBG|VBN|VBP|VBZ)(?!\b(w+)\b))/g) {
9         $verbs{$1}++;
10     }
11 }
12
13
14 my @top20 = (sort { $verbs{$b} <=> $verbs{$a} } keys %verbs)[0..19];
15
16 print "Verb Count:\n";
17
18 foreach my $verb (@top20) {
19     print "$verb\t$verbs{$verb}\n";
20 }
```

The terminal at the bottom shows the command `perl question_7.pl HW4_wsj_pos.txt/wsj_pos.txt` and its output:

```
rthompson@DESKTOP-EAHC8Q0:~/arizona/hlt538-computational_linguistics/hw4$ perl question_7.pl HW4_wsj_pos.txt/wsj_pos.txt
says      1872
had        941
were       904
been       852
s          593
do         479
say        401
make       317
did        310
made       303
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.