Department of Mathematics and Computing Science CSCI 3430 - Principles of Programming Languages

Assignment #2

Question 1

In order to verify the regular expressions, firstly, create a txt file named test.txt with the content as below:

```
123
1234
124
c234s
that
THAT
TTHAT
Here 12
HERE123
THERE
11here
THIS
THAT
THIS THAT THE OTHER THING
THIS THAT
mike,lee
Mike,Lee
~
```

a. All lines exactly 3 characters long

grep '^...$' test.txt

After the above command, the result is as below:

```
$ grep '^...$' test.txt
123
124
```

b. All lines starting with “c”, ending with “s” and exactly 5 characters long

grep '^c...s$' test.txt

```
$ grep '^c...s$' test.txt
c234s
```

c. All lines that DO NOT contain the word “THAT” (case insensitive)

grep -i -v 'THAT' test.txt

```
$ grep -i -v 'THAT' test.txt
123
1234
124
c234s
Here 12
HERE123
THERE
11here
THIS
mike,lee
Mike,Lee
```

d. All lines that contain the word (space delimited) “Here” but not “There” (case sensitive)

grep -v -w 'There' test.txt |grep -w 'Here'

```
$ grep -v -w 'There' test.txt |grep -w 'Here'
Here 12
```

e. All lines that contain “THIS” and “THAT” but not “THE OTHER THING”

grep 'THIS' test.txt | grep 'THAT' |grep -v 'THE OTHER THING'

```
$ grep 'THIS' test.txt  | grep 'THAT' |grep -v 'THE OTHER THING'
THIS THAT
```

f. An 8 character name, where the first character must be an uppercase letter (not a number), the rest can be letters or numbers, with no internal punctuation, followed by a comma”,” followed by the last name. Since first and last name have the same syntax it’s a great opportunity to reuse syntax.

grep '^[A-Z]\w\{0,7\},[A-Z]\w\{0,7\}' test.txt

```
$ grep '^[A-Z]\w\{0,7\},[A-Z]\w\{0,7\}' test.txt
Mike,Lee
```

Question 2

Fix up the example grammar included below (shown below as taken from the slides) to include / and * operators with correct precedence and allow an unlimited number of <term>. Make sure the grammar only evaluates one way, grouping / before * and before either + or -.

<expr> ::= <term> + <term> | <term> - <term>

<term> ::= <var> | const

<var> ::= a | b | c | d | e

Solutions:

<expr> ::= <expr> + <exprD> | <exprD> + <expr> | <expr> - <exprD> | <exprD> - <expr> | <exprD>

<exprD> ::= <exprD> / <exprM> | <exprM> / <exprD> | <exprM>

<exprM> ::= <exprM> * <term> | <term>

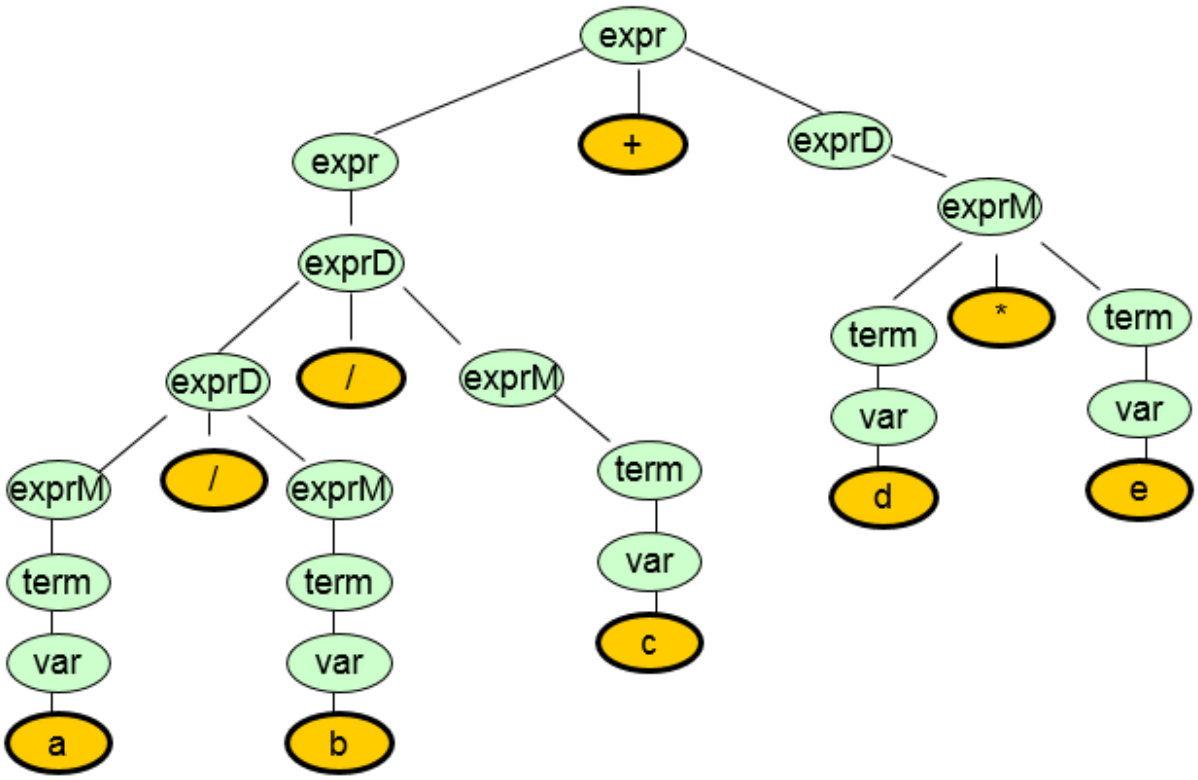
<term> ::= <var> | const

<var> ::= a | b | c | d | e

Question 3

Draw a parse tree using the rules you created in question 2 for the expression a/b/c+d*e using ASCII Art or some drawing package like Visio, MS PowerPoint, MS Word or even good old MS paint.

Solutions:



Question 4

Write a set of grammar rules that recognizes the following URLs (valid characters, http:// prefix, arbitrary number of “.” And “?” and “/” etc). You do not need an exhaustive thing to match ALL possible URLS, just restrict this to a two systems in the following (note that some have trailing slashes and some do not)

Solutions:

<url> ::= <http> <hostname> [<folders>] [<page>]

<http> ::= 'http://'

<hostname> ::= <word>.<hostname> | <word>.<word>

<folders> ::= /~<word> | /<word>/<folders> | /<word>

<page> ::= /<file>[<link>]

<file> ::= <word> | <word>.<word>

<link> ::= ?<word>=<digit>

<word> ::= {char}

<digit> ::= {number}

<number> ::= 0|1|2|3|4|5|6|7|8|9

<char> ::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z