Regular Expressions (cont.)

- **Special characters** in regular expressions:
  - .: matches almost any character except line breaks
  - a.e: water, ate, gate
  - *: match the prior expression zero or more times
  - \.: decimal point
  - How do we write a regex to match the floating point values with one digit after the decimal points. \[d*\d: .3, 12.5, 139.9\]
  - -: range indicator
  - [a-z]: one lowercase letter from a to z
  - ^: negates an expression when inside brackets, permits you to specify strings that don’t include a certain expression, or is the start of the string if it is outside of the brackets.
    - *[0-9]: matches any character that is not a digit
    - ^a: matches strings start with a
  - $: the end of the string
    - the end$: this is the end
    - How do we write a regex that can match any number between 1000 and 9999?
      - ^[1-9][0-9]{3}$
      - create a text.txt with following numbers, one on each line, 1231 21 5 57 01001 100001 1000a; key in the command `egrep "^[1-9][0-9]{3}$" text.txt`; this command returns the matches in text.txt.
  - (): group tokens (different from EBNF)
    - th(e | is): the, this
  - +: match the prior expression one or more times
    - html tags: `<html> </html>, <h1></h1>, <div id="block1"></div>
    - `<[^< >]+>`: matches HTML tags without regard to attributes
  - \{min, max\}: specify how many times a token can be repeated, min >=0 minimum number of matches, max >= min maximum number of matches. If \{min, \} the maximum number of matches is infinite. \{min\} repeat exactly min times.
    - {0, } same as *, {1, } same as +
    - ^[1-9][0-9]{3}$: matches a number between 1000 and 9999
    - ^[1-9][0-9]{2,4}$: matches a number between 100 and 9999
Abstract Syntax Tree
- It’s also worth noting that the output of syntactic analyzer is an abstract syntax tree.
- So, what is an abstract syntax tree? We are going to use Clite to illustrate it.
- Before moving forward to the details of abstract syntax tree, let’s take a look at Clite syntax first.
- Clite is a subset of C language. Its syntax is given below.
  - {} and [] in bold are EBNF meta-symbols.

Concrete Syntax of Clite (from Tucker and Noonan, 2007).

```
Program -> int main ( ) { Declarations Statements }
Declarations -> { Declaration }
Declaration -> Type Identifier [ [ Integer ]] { , Identifier [ [ Integer ]] }
  Type -> int | bool | float | char
Statements -> { Statement }
Statement -> ; | Block | Assignment | IfStatement | WhileStatement
  Block -> { Statements }
Assignment -> Identifier [ [ Expression ] ] = Expression;
IfStatement -> if ( Expression ) Statement [ else Statement ]
WhileStatement -> while ( Expression ) Statement

Expression -> Conjunction { | Conjunction }
Conjunction -> Equality { & & Equality }
  
Equality -> Relation [ EquOp Relation ]
  EquOp -> == | !=
Relation -> Addition [ RelOp Addition ]
  RelOp -> < | <= | > | >=
Addition -> Term { AddOp Term }
  AddOp -> + | -
Term -> Factor { MulOp Factor }
  MulOp -> * | / | %
Factor -> [ UnaryOp ] Primary
  UnaryOp -> - | !
Primary -> Identifier [ [ Expression ] ] | Literal |
  ( Expression ) | Type ( Expression )

Identifier -> Letter { Letter | Digit }
  Letter -> a | b | ... | z | A | B | ... | Z
Digit -> 0 | ... | 9
Literal -> Integer | Boolean | Float | Char
Integer -> Digit { Digit }
Boolean -> true | false
  Float -> Integer . Integer
Char -> ' ASCIIChar '
```
Read Clite syntax, and answer the following questions.

- What are the T, N, P, S in Clite?
  - T: A ~ Z, a ~ z, 0 ~ 9, ASCIIChar, true, false, int, bool, float, char, main, if, while, operators
  - N: the symbols on the left-hand-size of the productions
  - P: all the rules
  - S: Program

- What is the precedence of operators?