Assignment 2 & 3

1. [10 points] Describe some strings that are represented by the following regular expressions. **NOTE:** each bullet point contains a *single* regular expression
   * -?[0-9]+(\\*10\^)?[1-9]\*
     + -3\*10^222
     + 111\*10^3
     + 8
   * [A-Z]+ and ([A-Z]+|\.\.\.)
     + COMP and SCI
     + COMP and ...
     + XYZ and A
2. [10 points] Build the regular expression for the following descriptions. Don’t forget to take the space character in description two into account. **NOTE:** This question *must* be submitted in a digital format (no scans)
   * Identifiers in a language that must start with an underscore character and must end in a numeric digit. The length can be any size and values in between can be any alphanumeric character.
     + (-+)\w\*[0-9]+
   * A phone number with either the following formats: (888) 888-8888 or 888-888-8888
     + (\([0-9]{3}\) [0-9]{3}-[0-9]{4})|([0-9]{3}-[0-9]{3}-[0-9]{4})
   * The VCU V Number
     + V[0-9]{8}

NOTE: For all following grammars, only the 🡪 | { } characters are valid metasymbols. If these characters are enclosed in single quotes, they should be treated as terminals and not metasymols. All other characters should be treated as terminals.

1. [10 points] Using the grammar below, provide a leftmost derivation for the following statement:   
   * if ( hello == 8 ) { world = 16 ^ 2 }
2. *if-statement* 🡪 if ( *condition* ) '{' *assign-stmt* '}'
3. *condition* 🡪 *var* == *int*
4. *assign-stmt* 🡪 *var* = *stmt*
5. *stmt* 🡪 *int* + *int* | *int* - *int* | *int* ^ *int*
6. *var* 🡪 hello | world
7. *int* 🡪 *int* *digit* | *digit*
8. *digit* 🡪 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

If-statement -> if (*condition*) ‘{‘ *assign-stmt* ‘}’

* if ( *var* == *int* ) ‘{‘ *assign-stmt* ‘}’
* if ( hello == *int*) ‘{‘ *assign-stmt* ‘}’
* if ( hello == 8 ) ‘{‘ *assign-stmt* ‘}’
* if ( hello == 8 ) ‘{‘ *var* = *stmt* ‘}’
* if ( hello == 8 ) ‘{‘ hello = *stmt* ‘}’
* if ( hello == 8 ) ‘{‘ hello = *int* ^ *int* ‘}’
* if ( hello == 8 ) ‘{‘ hello = 16 ^ *int* ‘}’
* if ( hello == 8 ) ‘{‘ hello = 16 ^ 2 ‘}’

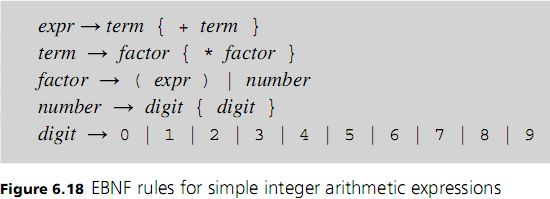
1. [10 points] Using the grammar below, provide a leftmost derivation for the following statements:
   * if ( myVar <= 5 ) { otherVar = 256 % 8 }

If-statement -> if (*condition*) ‘{‘ *assign-stmt* ‘}’

* if ( *var* <= *int* ) ‘{‘ *assign-stmt* ‘}’
* if ( myVar <= *int*) ‘{‘ *assign-stmt* ‘}’
* if ( myVar <= 5 ) ‘{‘ *assign-stmt* ‘}’
* if ( myVar <= 5 ) ‘{‘ *var* = *stmt* ‘}’
* if ( myVar <= 5 ) ‘{‘ otherVar = *stmt* ‘}’
* if ( myVar <= 5 ) ‘{‘ otherVar = *int* % *int* ‘}’
* if ( myVar <= 5 ) ‘{‘ otherVar = 256 % *int* ‘}’
* if ( myVar <= 5 ) ‘{‘ otherVar = 256 % 8 ‘}’

1. *if-statement* 🡪 if (c*ondition*) '{' *assign-stmt* '}'
2. *condition* 🡪 *var* == | <= | >= | < | > *int*
3. *assign-stmt* 🡪 *var* = *stmt*
4. *stmt* 🡪 *int* [+\-\*\/%^] *int*
5. *var* 🡪 ?[a-zA-Z]+
6. *int* 🡪 [0-9]+
7. [10 points] Using the grammar below, how many legal sentences are there? Why is that? Suppose white space was completely ignored in the grammar so that sentences could be written as “thecompetitorseesawin.” Can this grammar still be parsed? Explain
8. sentence 🡪 noun-phrase verb-phrase
9. noun-phrase 🡪 article noun
10. article 🡪 a | and | the
11. noun 🡪 girl | competitor | win | dog | comp
12. verb-phrase 🡪 verb noun-phrase
13. verb 🡪 sees | permits | objects

300 sentences can be made. The words can be combined in such a way that there are 300 unique sentences. If whitespace is ignored, this grammar could not be parsed. The whitespace is used as the delimiter.

1. [20 points] Add the following four operations in the proper location for the order of operations to apply to the EBNF grammar below. Please provide a complete EBNF grammar for your answer.
   * subtraction,
   * division,
   * integer modulus division,
   * exponents

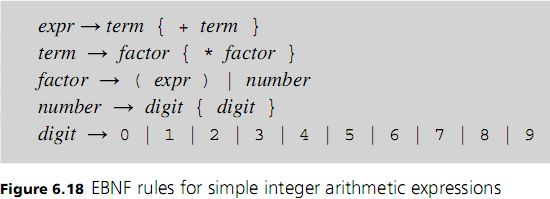
*expr* -> *term* { + | - *term* }

*term* -> *factor* { \* | / | % *factor* }

*factor* -> factor { ^ factor }

*number* -> *digit* { *digit* }

*digit* -> 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

1. [20 points] Using the grammar below, draw the following:
   * (8 \* 3 \* (4 + 5)) – parse tree
   * (4 + 5) + 16 + (3 \* 9) – abstract syntax tree

See attached photos

1. [20 points] Using the grammar and statement below, draw a parse tree, abstract syntax tree, and all intermediate steps for converting the parse tree to the abstract syntax tree.
   * if ( myVar <= 5 ) { otherVar = 256 % 8 }
2. *if-statement* 🡪 if ( *condition* ) '{' *assign-stmt* '}'
3. *condition* 🡪 *var* == | <= | >= | < | > *int*
4. *assign-stmt* 🡪 *var* = *stmt*
5. *stmt* 🡪 *int* [+\-\*\/%^] *int*
6. *var* 🡪 [a-zA-Z]+
7. *int* 🡪 [0-9]+