Stamatios Morellas
COM S 342 - Homework 1
Due 1/31/20 @ 11:59pm

1. True/False. Justify your answer.

a) Two or more grammars may generate the same language - **TRUE**
b) Ambiguity (in the meaning of a grammatically valid sentence) implies that the grammar, from which the sentence is generated, is ambiguous - **TRUE**
c) A language is ambiguous if there is some ambiguous grammar that generates it - **FALSE**
d) Any palindrome over the letters a and b can be generated by the following grammar:
$S \rightarrow aSa \mid bSb \mid \varepsilon$ - **TRUE**
e) The result of the Java expression

```
++y + ++x * 2
```

is 53 when y is 10 and x is 20. - **TRUE**

2. Consider the following grammars and order them as per > relation as described in the problem. Justify your answer.

$G1: S \rightarrow a \mid b \mid aSa \mid bSb \mid \varepsilon$
$G2: S \rightarrow a \mid b \mid Sa \mid Sb \mid \varepsilon$
$G3: S \rightarrow a \mid b \mid aa \mid bb \mid aSa \mid bSb \mid \varepsilon$

(ε denotes empty string)

G2 > **G1** – Strings generated by G1 can be generated by G2, but not all strings generated by G2 can also be generated by G1
G3 > **G1** – Strings generated by G1 can be generated by G3, but not all strings generated by G3 can also be generated by G1
G2 > **G3** – Strings generated by G3 can be generated by G2, but not all strings generated by G2 can also be generated by G3

3. Consider the following grammar for expressions:

$$E \rightarrow E + E \mid E/T \mid T$$
$$T \rightarrow T - R \mid T * R \mid R$$
$$R \rightarrow Number$$

a) Is the above grammar ambiguous? Justify your answer.

Yes it is; we can create two different parse trees with the same sentence as shown below. Since ambiguity occurs when there are at least two different parse trees resulting from the derivation of the same sentence.

$E \rightarrow E + E$ $\rightarrow T + E$ $\rightarrow R + E$ $\rightarrow 1 + E$ $\rightarrow 1 + T$ $\rightarrow 1 + T * R$ $\rightarrow 1 + T * 3$ $\rightarrow 1 + R * 3$ $\rightarrow 1 + 2 * 3$	$E \rightarrow E + E$ $\rightarrow T + T$ $\rightarrow R + T$ $\rightarrow 1 + T$ $\rightarrow 1 + T * R$ $\rightarrow 1 + T * 3$ $\rightarrow 1 + R * 3$ $\rightarrow 1 + 2 * 3$

b) What is the associativity and precedence-order of the operators: $+$, $-$, $*$, / ? Justify your answer.

Since each of these operators are read from left to right, this means that their associativity is also left to right. The division and multiplication operators have the highest precedence-order, followed by addition and subtraction, as these operations are handled before addition and subtraction due to the order of operations.

c) Draw a derivation sequence and corresponding parse trees for the following expressions (assume Number denotes any integer

i) 1 + 2 -3 - 4

ii) 1 + 2 * 3 /4 / 5 - 6

i)	ii
$E \rightarrow E + E$ $\rightarrow T + T - R$ $\rightarrow T + (T - R) - R$ $\rightarrow 1 + (2 - 3) - 4$	$E \rightarrow E + E$ $\rightarrow T + E/T$ $\rightarrow R + (E/T) / (T - R)$ $\rightarrow R + T * R / T / T - R$ $\rightarrow 1 + 2 * 3 / 4 / 5 - 6$

4. We plan to define a new programming language. A statement in this language can be of the following form:

- an assignment statement var := E, where E is an arith-expression involving variables, integers and some arithmetic operators;
- a conditional statement `if BE then statement` where BE is a boolean expression representing relationship between two arith-expressions.
- unconditional statement goto id where id is a label of some statement
- a statement may have a label—id statement, where id denotes the label of statement
- a compound statement which starts with a begin and ends with an end and between these delimiters contains at least two statements separated by ; (i.e., the last statement is not preceded by ;).

You can assume typical infix grammar for any arithmetic expression and boolean expressions: for instance x+1 is an arithmetic expression x<20 is a boolean expression.

a) Does the following program satisfy the above rules? Justify your answer.

```
L0 : begin
L1 : x := x + 1
L2 : if x < 20 then goto L1;
L3 : y := x
end
```

L0: The **fifth** rule is satisfied by this program because there are at least 2 statements in between the 'begin' and 'end' statements.

L1: This line satisfies the **first** rule, which states that E, as stated in the expression 'var := E', is an arithmetic expression with variables, integers, and operators.

L2: This line satisfies the **second** and **third** rules since the conditional statement in the program has the correct formatting, as well as the 'goto' statement'.

L3: Every line has a label, so the **fourth** rule is also satisfied.

Therefore, this program **does** satisfy all the rules.

b) Does the following grammar (start symbol is P) correctly represents the above rules? That is, if a program follows the rules then it is also generated by the grammar; and if a program is generated by the grammar then it also follows the rules. Justify your answer.

$P \rightarrow L\ St$

$L \rightarrow id$

$$St \rightarrow Var := E \mid if\ BE\ then\ St \mid goto\ L \mid begin\ StSeq\ end$$
$$StSeq \rightarrow St \mid St;\ StSeq$$

Assume that the grammar for variable names is denoted by Var, the grammar for type of labels is denoted by id, grammar for all possible arith-expressions and all possible boolean expressions as per the rules are given by E and BE, respectively.

The grammar shown above satisfies all of the rules listed in the description. Each statement is either an assignment statement; a conditional statement `if BE then Statement,` where BE is a boolean expression; an unconditional statement `goto id`; or a compound statement that starts with a `begin` and ends with an `end`, and has at least 2 statements in-between the two.