CSCI 4220
– Assignment 2 –

Introduction

Your solutions to this assignment should be submitted to Canvas as an appropriately named PDF file (e.g., my_name_A2.pdf). Note that solutions to many of these problems are posted. It is your responsibility to check your solutions with the ones posted. Feel free to contact me with any questions you might have. Especially, if there is something in a posted solution that you do not understand.

IMPORTANT: Also, solutions should NOT be hand written (e.g., a hand written parse tree that is then scanned and pasted into a document). Parse trees should be created using tools like PowerPoint.

Turing Completeness

Both language 1 and 2 (shown below) are **Turing complete** in the sense that they can be used to describe all the computations that a Turning machine can perform.

Language 1	**Language 2**
boolean expressions	boolean expressions
integer variables	integer values
integer arithmetic	integer arithmetic
(including ++ operators like those found in C++)	function definition
sequential execution of statements	function call
if-then statements	conditional-expressions
assignment statements	recursion
while-loops	

Problem 1 *(20 points)*
Prove that a language is also Turing complete if it (just) contains the following constructs:

boolean expressions
integer variables
integer arithmetic
sequential execution of statements
assignment statements
while-loops

Note that "proof by specific example" is not a valid technique here. (Hint: Language equivalence). Note that the "break" command is not part of any of these languages.

BNF Grammars and Regular Expressions

Problem 2 *(10 points)*

Give a BNF grammar that approximates the syntactic structure of a textbook. The description should be quite detailed but need not be exact. An example of an English approximation of the structure of a book might begin as follows: A book has a cover page containing the title. After the cover page, there is a table of contents containing chapter titles as well as starting page numbers. Then comes a preface which consists of several paragraphs of text. Note that the preface is part of the contents. Chapters consist of sections that contain one or more paragraphs. Each chapter must end with an exercises section. Paragraphs consist of one or more sentences, etc. etc.

After constructing your grammar, describe five areas where the grammar fails to precisely capture the structure of the book. For example, if your description allows a book to contain an arbitrary number of chapters, then the grammar will not be able to enforce that chapters are numbered consecutively. For example, the chapter sequence 1, 3, 2 would be considered to be syntactically legal according to the grammar.

Problem 3 *(20 points)*

Give an unambiguous BNF grammar (do not use an extended BNF) describing a language of balanced parenthesis. Note that a left parenthesis must appear to the left of its corresponding right parenthesis in order to be considered balanced. From this it follows that ")(" does not constitute a string of balanced parenthesis. Also, this language does not contain the empty string, and the only two tokens in the language are (and). For example, ()() and ((()))()(()()))) are in this language, while (((() is not.

Problem 4 *(10 points)*

Give an unambiguous BNF grammar (do not use an extended BNF) for $\{0^i 1^i \mid i \geq 0\}$. *Note that, the expression* 0^i *denotes* i *repetitions of the symbol* 0. *For example,* $0^3 1^3$ *denotes* 000111. $L = \{\epsilon, 01, 0011, 000111, ...\}$.

Problem 5 *(15 points)*

Consider the following grammar:

$A ::= A\ b\ A \mid c\ A\ A \mid B \mid d$
$B ::= b\ A\ A \mid c$

Can the following string be generated by A?

$$c\ d\ d\ b\ b\ c\ d\ d\ d$$

*If your answer is "yes", draw a properly formed parse tree whose leaves whose root is A and whose leaves are: c d d b b c d d d . **Note:** To receive credit for a "yes" answer you **must** draw a parse tree. I suggest that you use either powerpoint or Mayura draw to generate this figure.*

Hint0: Work incrementally.
Hint1: Consider constructing a parse tree from the bottom up as well as top down.

Problem 6 *(10 points)*

You are given the following alphabet: { a, b, c, d}.

1. *Write a regular expression describing the set of all identifiers (i.e., strings) having at most 2 occurrences of the letter c. For example, the identifier "abbcdaac" would belong to this set, but the identifier "aabcdaacc" would not.*

2. *Write a regular expression describing the set of all identifiers having at least 2 occurrences of the letter c.*

Problem 7 *(15 points)*

Consider two definitions of the language of mathematical expressions. This language contains the following operators and constants:

- *Arithmetic Operators and their signatures.*

Name of Operator	*Operator Symbol*		*Type Expression*
addition	$+$	:	*integer * integer* $\rightarrow$ *integer*
subtraction	$-$	:	*integer * integer* $\rightarrow$ *integer*
multiplication	$*$	:	*integer * integer* $\rightarrow$ *integer*
division	$/$	:	*integer * integer* $\rightarrow$ *integer*
exponentiation	$**$	:	*integer * integer* $\rightarrow$ *integer*

Note that the signature of an operator is an expression of the form $f : \tau$ where f is the symbol denoting the operator and τ is a type expression that describes the types of the operands of f and the type of its result. For example, integer addition is a binary operator that takes two integers as its operands and produces an integer as its result. Formally, we will write the signature of addition as follows:

$$+ : integer * integer \rightarrow integer$$

In the expression integer $*$ integer the $*$ denotes a domain cross-product. In particular, in this context $*$ does ***NOT*** *denote multiplication.*

- *Constants*
  - $0, 1, 2, 3, ...$

Grammar 1

expr	::=	*expr1* | *expr + expr1* | *expr - expr1*
expr1	::=	*expr2* | *expr1 * expr2* | *expr1 / expr2*
expr2	::=	*integer* | *- integer* | *expr3*
expr3	::=	*integer ** expr2*

Grammar 2

expr	::=	*expr1* | *expr1 + expr* | *expr1 - expr*
expr1	::=	*expr2* | *expr2 * expr1* | *expr2 / expr1*
expr2	::=	*expr2 ** expr3* | *expr3*
expr3	::=	*integer* | *- integer*

1. *Using both grammars, draw a parse tree for the expression:* $4 - 2 - 1 * *2 * 5$
2. *What is the difference between these two grammars?*
3. *Describe the practical significance/impact of this difference and give arguments in favor of choosing one grammar over the other.*