BBM 301 - Programming Languages - Fall 2020 Midterm
December 1, 2020 – Part 1

Name: _____

Student ID number: _____

Please write your name, ID and following honor pledge:
"**On my honor, I pledge that I have neither given nor received any unauthorized assistance on this exam.** "
and **sign** your answer sheet.

Question 1-2: 23 points, 20 mins.

1- [8 pts] A natural number is either 0 or any positive number consisting of digits 0-9, but that does not start with 0s. That is, the numbers 0, 9, 78, 304, 40105, 20000000 are natural, but 02 and 0034 are not.

a) Write a BNF for describing natural numbers. Do not use EBNF.

b) Write a regular expression for describing natural numbers

2- [15 pts] Consider the following grammar, where {S} is the set of non-terminals, and {x; &} is the set of terminals.

```
S -> x
S -> S &
S -> S S
```

a) Show that this grammar is ambiguous.

b) Write a BNF for non-ambiguous grammar for the same language, resolving ambiguities as follows:
- Juxtaposition (the production `S -> S S`) associates to the left, & is non-associative
- & has a higher precedence than Juxtaposition.

BBM 301 - Programming Languages - Fall 2020 Midterm
December 1, 2020 – Part 2

Name: _____

Student ID number: _____

Please write your name, ID and following honor pledge:
"**On my honor, I pledge that I have neither given nor received any unauthorized assistance on this exam.** "
and **sign** your answer sheet.

Question 3 - 32 points, 25 mins.

3- BNF for simple Scheme

a) **[10 points]** A list in a functional programming language is either an empty list or a list of elements which are either atoms or other lists. A list is enclosed in parentheses and elements of the list are separated with spaces. Assume that atoms are represented by single upper case letters. Some possible lists are

```
(P Q)
((X Y) Z)
()
(((A B C) (D E)) F (G H))
```

Write a grammar using the BNF to describe such lists. Do not use EBNF notation.

b) **[22 points]** Write rules in BNF to describe a grammar for a simplified Scheme that includes only the limited set of constructs listed below. Do not use EBNF notation.

- numbers
- names
- parameters
- primitive numeric functions : +, −, *, /, MIN, MAX, and SQRT.
- function QUOTE
- list functions CAR and CDR
- lambda expressions
- function definition using DEFINE

Be sure that your grammar accepts the following example expressions:

```
(+ 3 4 5)
(- 20 (* 4 2))
(MIN 5 3 4)
(LAMBDA (x) (+ x 1))
(LAMBDA (a b c) (+ (* a a) (* b b) c))
(DEFINE pi 3.14)
(DEFINE fun (LAMBDA (x) (+ x 1)))
(QUOTE (1 2 3))
(CAR (QUOTE (2 3 4)))
(CDR (QUOTE (2 3 4)))
(DEFINE hypotenuse (LAMBDA (a b) (SQRT (+ (* a a) (* b b))) ) )
```

```
(DEFINE second (LAMBDA (lst) (CAR (CDR lst)) ) )
```

BBM 301 - Programming Languages - Fall 2020 Midterm
December 1, 2020 – Part 3

Name: _____

Student ID number: _____

Please write your name, ID and following honor pledge:
"**On my honor, I pledge that I have neither given nor received any unauthorized assistance on this exam.** "
and **sign** your answer sheet.

Question 4: 20 points, 20 mins.

Here is a simple grammar for very limited programming language:

<program> → **begin** <stmt-list> **end**
<stmt-list> → <stmt-list> **;** <stmt>
| <stmt>
<stmt> → **for** <id> **=** <val> **to** <val> **{** <stmt-list> **}**
| **print hello**

In this grammar, non-terminals are shown in <> and terminals are shown in lower bold case. <id> represents a variable name that can consist of numbers and characters, <val> represents an integer value.

Write the **lex and yacc files** for the interpreter that will compute the number of *print hello* commands that will be executed. Note: Use yacc's expression value assignment features ($$'s).

BBM 301 - Programming Languages - Fall 2020 Midterm
December 1, 2020 – Part 3

Name: _____

Student ID number: _____

Please write your name, ID and following honor pledge:
"**On my honor, I pledge that I have neither given nor received any unauthorized assistance on this exam.** "
and **sign** your answer sheet.

Question 5: 25 points, 25 mins.

a) [13 points] Write a Scheme function named `checkOrderedPairs` which takes a predicate function and a simple list as parameters, and returns the number of pairs that satisfy the given predicate function. The predicate must accept two elements as its arguments and should process the pairs according to their order. Note that you should only process the pairs in given order, for example, if the input is (3 5 7 8), the pairs to process will be (3,5),(3 7),(3 8),(5,7),(5,8) and (7,8).

Here are some example calls and the corresponding outputs:

```
> (checkOrderedPairs equal? '(a b c b a))
2
> (checkOrderedPairs equal? '(a b c d e))
0
> (checkOrderedPairs < '(2 16 5 8 4))
5
```

b) [12 points] Write a Scheme function named `countEvenNumbers`, which counts the even numbers at every level of a given list. Here are some example calls and their outputs:

```
> (countEvenNumbers '())
0
> (countEvenNumbers '(2 5))
1
> (countEvenNumbers '(1 2 (4 8) 10))
4
> (countEvenNumbers '(1 3 (a b (4 5 (6)))))
2
```