Spring 2018 CSC 419 - Programming Languages - Assignment 2

Due date: Tuesday, Feb 13, 2018, by midnight, upload in home folder of class

Please put your full name on the assignments!

All your files for an assignment should be in a subdirectory in your home directory. The subdirectory should be named "assignmentX' where X is the assignment number. The TA will pick up the files from this subdirectory after the deadline for the assignment.

Exercise 2.1

Write EBNF descriptions (and also show one example) for the following:
[6 points]

a) A **switch** statement in C or Java

b) C **float** literals

c) A **function** definition in C or C++

Exercise 2.2*

a) Given the following EBNF declarations (based on Modula-2)

<simple expression> => [+ | -] <term> { + <term> | - <term> | OR <term> }
<term> => <factor> { * <factor> | / <factor> | DIV <factor> | MOD <factor> | AND <factor> | & <factor> }
<factor> => <number> | <ident> | (<simple expression>) | NOT<factor>
<ident> => <letter> { <letter> | <digit> }

Simplify the expressions for <simple expression> and <term> by adding other nonterminal definitions so that you can reduce the number of options in the braces.
[2 points]

b) Construct five examples of a <simple expression>, using only terminals in the language. Try to make each example significantly different from the others so as to illustrate the range of expressions which this definition encompasses. Assume simple definitions for <letter> (i.e., a..z, A..Z), <digit> (i.e., 0..9), and <number> (i.e., <digit>{<digit>}) although the actual definitions in Modula-2 are more complex.
[5 points]

Exercise 2.3*

a) Here are more EBNF declarations:

<case statement> => CASE <expression> OF <case> { '|' <case> } [ELSE <statement sequence>] END
<case> => [<case label list> ':' <statement sequence>]
<case label list> => <case label> { ',' <case label> }
<case label> => <const expr> ['..' <const expr>]

According to the EBNF definitions given in this exercise, indicate whether each of the following examples is or is not a legal case statement. For those that are not, explain why they are illegal according to the grammar. Assume that integers and single characters en-

closed in single quotation marks are constant expressions (i.e., <const expr>), that a variable name qualifies as an <expression>, that a <statement sequence> is a sequence of semicolon-separated statements, and that a procedure call (indicated by a simple identifier, not followed by parentheses) and an assignment statement (where'=>' is the assignment operator) are examples of statements. Also, 'DIV' is a legal division operator in this example language (Modula-2).
[7 points]

1.
```
CASE nextChar OF
  'a' : ScanIdentifier |
  'b' : ScanIdentifier;
        bold := 1 |
  '0' : ScanNumber
ELSE Error
END
```

2.
```
CASE nextChar OF
  'a' .. 'z' : ScanIdentifier
ELSE
  '0' .. '1' : ScanNumber
END
```

3.
```
CASE nextChar OF
  'a' .. 'z' : ScanIdentifier
ELSE
  ScanNumber
ELSE
   ScanOperator
END
```

4.
```
CASE nextChar OF
  'a' .. 'z', 'A' .. 'Z' : ScanIdentifier |
  '0' .. '9' : ScanNumber |
  '+', '-', '*', '/', '^' : ScanOperator |
END
```

5.
```
CASE nextChar OF
  'a', 'b', 'c', .. 'z' : ScanIdentifier |
  '0' .. '9' : ScanNumber |
  '+', '-', '*', '/', '^' : ScanOperator
END
```

6.
```
CASE op OF
  | '+' : total := total + operand
  | '-' : total := total - operand
  | '*' : total := total * operand
  | '/' : total := total DIV operand
END
```

7.
```
CASE ident OF
END
```

*both exercises have been adopted from Jonathan Mohr