Coursework 2

6CCS3CFL - Compilers & Formal Languages

Finley Warman

November 3, 2020

N.B. to run all test cases and questions, run: '`amm 02_coursework.sc all`'

Contents

Question 1

Implemented 'simple' and 'extended' regular expressions, including REC, in `02_coursework.sc`. Expressions added for all tokens described, including expression `WHILE_LANG_REG` representing all valid tokens of the WHILE language.
(See KEYWORD, OP, ID, SEMI, NUMBER, COMMENT, STRING, LPAREN, RPAREN, LCURL, RCURL, WHITESPACE)

Question 2

Q: Definitions for mkeps:

```
...
mkeps([c1,...,cn]) == N/A, not nullable
mkeps(r+)          == Plus[mkeps(r)]
mkeps(r?)          == Empty
mkeps(r{n})        == if (n=0) NTimes[] else NTimes[mkeps(r)]
```

Q: Definitions for inj:

```
...
inj([c1,...,cn]) c Empty        == Char(c)
inj(r+) c Seq(v, Stars vs)      == Plus(inj(r) c v :: vs)
inj(r?) c v                     == Opt(inj(r) c v)
inj(r{0}) c v                   == Empty
inj(r{n}) c Seq(v, NTimes vs)   == NTimes(inj(r) c v :: vs)
```

To test $a^{\{3\}}$ and $(a + \mathbf{1})^{\{3\}}$, call `amm 02_coursework.sc lex_extended`.

Tokens for expressions from Q1:

```
WHILE_LANG_REG = (
  ("kwd" : KEYWORD) +
  ("op"  : OP)      +
  ("id"  : ID)      +
  ("semi": SEMI)    +
  ("num" : NUMBER)  +
  ("comm": COMMENT) +
  ("str" : STRING)  +
  ("brkt": (LPAREN + RPAREN)) +
  ("crly": (LCURL  + RCURL )) +
  ("wspc": WHITESPACE)
)*
```

The token sequence for `read n;` (`amm 02_coursework.sc while_lang_lexing_tests`):

```
kwd:     "read"
wspc:    " "
id:      "n"
semi:    ";"
```

Question 3

Resulting tokens for each program, filtering whitespace `wspc`:
(`amm 02_coursework.sc program_lexing_tests`)

```
========== collatz.while ==========

kwd:     "write"
str:     "\"Input a number \""
semi:    ";"
kwd:     "read"
id:      "n"
semi:    ";"
kwd:     "while"
id:      "n"
op:      ">"
num:     "1"
kwd:     "do"
crly:    "{"
kwd:     "if"
id:      "n"
op:      "%"
num:     "2"
op:      "=="
num:     "0"
kwd:     "then"
id:      "n"
op:      ":="
id:      "n"
op:      "/"
num:     "2"
kwd:     "else"
id:      "n"
op:      ":="
num:     "3"
op:      "*"
id:      "n"
op:      "+"
num:     "1"
semi:    ";"
crly:    "}"
semi:    ";"
kwd:     "write"
str:     "\"Yes\""
semi:    ";"

==================================
```

```
========== factors.while ==========

comm:    "// Find all factors of a given input number\n"
comm:    "// by J.R. Cordy August 2005\n"
kwd:     "write"
str:     "\"Input n please\""
semi:    ";"
kwd:     "read"
id:      "n"
semi:    ";"
kwd:     "write"
str:     "\"The factors of n are\""
semi:    ";"
id:      "f"
op:      ":="
num:     "2"
semi:    ";"
kwd:     "while"
id:      "n"
op:      "!="
num:     "1"
kwd:     "do"
crly:    "{"
kwd:     "while"
brkt:    "("
id:      "n"
op:      "/"
id:      "f"
brkt:    ")"
op:      "*"
id:      "f"
op:      "=="
id:      "n"
kwd:     "do"
crly:    "{"
kwd:     "write"
id:      "f"
semi:    ";"
id:      "n"
op:      ":="
id:      "n"
op:      "/"
id:      "f"
crly:    "}"
semi:    ";"
id:      "f"
op:      ":="
id:      "f"
op:      "+"
```

```
num:      "1"
crly:     "}"

==================================
```

```
==========  loops.while  ==========

id:       "start"
op:       ":="
num:      "1000"
semi:     ";"
id:       "x"
op:       ":="
id:       "start"
semi:     ";"
id:       "y"
op:       ":="
id:       "start"
semi:     ";"
id:       "z"
op:       ":="
id:       "start"
semi:     ";"
kwd:      "while"
num:      "0"
op:       "<"
id:       "x"
kwd:      "do"
crly:     "{"
kwd:      "while"
num:      "0"
op:       "<"
id:       "y"
kwd:      "do"
crly:     "{"
kwd:      "while"
num:      "0"
op:       "<"
id:       "z"
kwd:      "do"
crly:     "{"
id:       "z"
op:       ":="
id:       "z"
op:       "-"
num:      "1"
crly:     "}"
semi:     ";"
id:       "z"
op:       ":="
id:       "start"
semi:     ";"
id:       "y"
op:       ":="
```

```
id:      "y"
op:      "-"
num:     "1"
crly:    "}"
semi:    ";"
id:      "y"
op:      ":="
id:      "start"
semi:    ";"
id:      "x"
op:      ":="
id:      "x"
op:      "-"
num:     "1"
crly:    "}"

==================================
```

```
========= collatz2.while ==========

comm:    "// Collatz series\n"
comm:    "//\n"
comm:    "// needs writing of strings and numbers; comments\n"
id:      "bnd"
op:      ":="
num:     "1"
semi:    ";"
kwd:     "while"
id:      "bnd"
op:      "<"
num:     "101"
kwd:     "do"
crly:    "{"
kwd:     "write"
id:      "bnd"
semi:    ";"
kwd:     "write"
str:     "\": \""
semi:    ";"
id:      "n"
op:      ":="
id:      "bnd"
semi:    ";"
id:      "cnt"
op:      ":="
num:     "0"
semi:    ";"
kwd:     "while"
id:      "n"
op:      ">"
num:     "1"
kwd:     "do"
crly:    "{"
kwd:     "write"
id:      "n"
semi:    ";"
kwd:     "write"
str:     "\",\""
semi:    ";"
kwd:     "if"
id:      "n"
op:      "%"
num:     "2"
op:      "=="
num:     "0"
kwd:     "then"
id:      "n"
```

```
op:      ":="
id:      "n"
op:      "/"
num:     "2"
kwd:     "else"
id:      "n"
op:      ":="
num:     "3"
op:      "*"
id:      "n"
op:      "+"
num:     "1"
semi:    ";"
id:      "cnt"
op:      ":="
id:      "cnt"
op:      "+"
num:     "1"
crly:    "}"
semi:    ";"
kwd:     "write"
str:     "\" => \""
semi:    ";"
kwd:     "write"
id:      "cnt"
semi:    ";"
kwd:     "write"
str:     "\"\\n\""
semi:    ";"
id:      "bnd"
op:      ":="
id:      "bnd"
op:      "+"
num:     "1"
crly:    "}"

==================================
```