Assignment 3

Microproject

Begin with the code from the class example (https://danielschlegel.org/wp/teaching/recursive-descent-parser-in-scala/) on recursive descent parsers in Scala. That program implements a parser for the following grammar:

```
1 S -: E$
2 E -: Terminal E2
3 E2 -: + E
4 E2 -: NIL
5 Terminal -: Const
6 Terminal -: Val
```

It also implements evaluation for parse trees in that grammar. Modify the program to implement the following, modified grammar:

```
1 S -: E$
2 E -: T E2
3 E2 -: + E
4 E2 -: NIL
5 T -: Terminal T2
6 T2 -: * T
7 T2 -: NIL
8 Terminal -: Const
9 Terminal -: Val
```

Be sure the parser works as well as the evaluation component. You will need to modify the case classes, the parser, and the evaluation.

Main Project

Write a Scala program that performs pattern matching on strings, where patterns are expressed using only the concatenation, alternation (“|”), and optional (“?”) operators of regular expressions (no loops/”*”, no escape characters), and the tokens are letters and digits, plus period (“.”) to mean any letter/digit. Each run of the program should accept a pattern, and then *any number of strings*, reporting only whether they match. Your program should represent expressions as trees (***use case***

classes) and evaluate on the inputs, **without using any regular expressions or Scala's regular expression library** except for matching the individual alphanumeric characters, if you'd like. For example:

```
pattern? ((h|j)ell. worl?d)|(42)
string? hello world
match
string? jello word
match
string? jelly word
match
string? 42
match
string? 24
no match
string? hello world42
no match
```

```
pattern? I (like|love|hate)( (cat|dog))? people
string? I like cat people
match
string? I love dog people
match
string? I hate people
match
string? I likelovehate people
no match
string? I people
no match
```

Bootstrap

An ambiguous grammar for patterns is:

```
S -: E$
E -: C | EE | E'|'E | E'?' | '(' E ')'
C -: '0' | '1' | ... | '9'| 'a' | 'b' | ... | 'z' | '.'
```

To reflect that option('?') has highest precedence, then concatenation, then alternation('|'), the following unambiguous grammar can be created:

```
S  -: E$
E  -: T '|' E | T
T  -: F T | F
F  -: A '?' | A
A  -: C | '(' E ')'
```

For the purposes of writing a recursive descent parser, this can be transformed into an ugly but simpler-to-use form:

```
  S  -: E$
  E  -: T E2
  E2 -: '|' E3
  E2 -: NIL
  E3 -: T E2
```

```
T  -: F T2
T2 -: F T2
T2 -: NIL
F  -: A F2
F2 -: '?' F2
F2 -: NIL
A  -: C
A  -: '(' A2
A2 -: E ')'
```

where '$' is eof/end-of-string, and NIL means empty (which in these productions means take the rhs only if others do not apply).

Use the following case classes to get you started. You will need several more, but you should be able to deduce the general pattern from these:

```
abstract class S
case class E(left: T, right: Option[E2]) extends S
case class E2(left: E3) extends S
case class E3(left: T, right: Option[E2]) extends S
```

You must implement a recursive descent parser yourself to build a tree of case classes from the input string. Remember not to use any regular expression processing other than for the individual characters (either built in or in external libraries)!

Extra Credit (+15%)

This project definitely includes some cases which are harder than others. One hard case which we'll count for extra credit is below:

```
pattern? a(b|bb)?bc
string? abc
match
string? abbc
match
string? abbbc
match
string? ab
no match
string? abbbbc
no match
```

Some Helpful Resources

Scala Pattern Matching Documentation (https://docs.scala-lang.org/tour/pattern-matching.html)

Suggested Development Environment

IntelliJ + Scala Plugin