Microproject

Write a Scala program which reads a string from the user, and uses either a recursive descent parser or parser combinators to determine if the input string matches the below grammar (i.e., is made up of a's and b's), building a parse tree along the way. Output the generated tree if the match is successful.

```
S -> E$
E -> C E2
E2 -> E
E2 -> NIL
C -> 'a' | 'b'
```

Read the below project description for assistance.

Main Project

Write a Scala program that performs pattern matching on strings, where patterns are expressed using only the concatenation, alternation ("|") optional ("?") operators of regular expressions (no loops/"*", no escape characters), and the tokens are letters and digits, plus period (".") to mean any letter. 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**. 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
```

Bootstrap

An ambiguous grammar for patterns is:

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

To reflect that option('?') has highest precedence, then concatentation, then alternation('|'), 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).

You may decide to implement a recursive descent parser yourself to build a tree from the input string, or use Scala's parser combinators to do the same (see Blackboard for a chapter on this topic). Remember not to use any regular expression processing (either built in or in external libraries)!

Note: As of Scala 2.11 parser combinators are in an external library, so you may need the jar file (https://mvnrepository.com/artifact/org.scala-lang/scala-parser-combinators/2.11.0-M4).