CSC 453 Milestone 4-5 (Grammar)

Goal:

These milestones are writing the EBNF grammar for Tau.

Specifications

The Tau language specification is a separate document.

The grammar should be in EBNF format. The format is described for the `rdgen` tool in the `rdgen` documentation.

Milestone 4 requirements

Milestone 4 requires that you specify the grammar for the entire Tau language **except** for the following:

- You may omit all unary and binary arithmetic and logical operators.
- You may omit all array syntax (type declarations and indexing).
- You may omit parenthetical expressions.

Milestone 5 requirements

Milestone 5 requires that you specify the grammar for the entire Tau language.

`rdgen` tool

You will need to install `rdgen` to generate the parser. `rdgen` is a Python program that generates a parser from an EBNF grammar.

You can `git clone` the `rdgen` tool from the `rdgen` repository on GitHub.

```
$ git clone git@github.com:proebsting/rdgen.git
```

The tool's instructions are in the README file at https://github.com/proebsting/rdgen

You will invoke `rdgen` with something like the following (depending on where you cloned `rdgen`):

```
python3 -m rdgen.main create --input grammar.ebnf --output parse.py
```

`parse.py`

DO NOT EDIT THE FILE GENERATED BY `rdgen`. (In subsequent milestones, you will be editing the file, but not these two.)

This file contains the parser, which is given by class `Parser`. It has two methods of interest:

- the constructor, which takes a scanner as an argument
- the `parse()` method, which parses the input.

Ambiguities

If your grammar contains LL(1) parsing ambiguities, you will need to resolve them for subsequent milestones. For these two milestones, it's OK if your parser works despite the ambiguities, although you can get a headstart on the next milestone by addressing them early.

Parsing ambiguities are signalled in the generated parser in comments that include the word "`AMBIGUITY`", which will include a terse explanation.

Notes

The Tau language specification is intentionally vague. Feel free to publically discuss the language specification on Piazza. If you have questions, please ask.

DO NOT, HOWEVER, GIVE GRAMMAR EXAMPLES ON PIAZZA. Instead, ask questions like, “Does the grammar allow ‘`x = 1 + 2`’? Why or why not?”

Difficulty

Your grammar may not be the same as my grammar. That’s OK. The goal is to get a working parser.

For reference, my grammar has 26 nonterminals and is 63 lines long with lots of blank lines. (Again, this is just for reference. Your grammar may be different.)

That said, grammar writing can be tricky. Start early and ask questions.

Standard Requirements

Your program must meet all the requirements outlined in the common requirements document.