CS 441
Spring 2025
Programming Assignment 1

For this assignment, you will use Racket, a functional programming language, to write a simple parser.

Note that we're writing only a parser, not a full interpreter (although interpreters in Racket aren't that difficult[1]). Your program only needs to produce a parse tree for the given input.

The standard interpreter for Racket is DrRacket, and is installed on all Flarsheim lab machines as well as available as a free download from https://racket-lang.org/.

Your code should have a function called `parse`, which takes one parameter—the name of the file of source code to be processed:

```
(parse "input.txt")
```

It should return a string: either "Accept", indicating the program is syntactically correct, and the parse tree; or a message as to which line the first syntax error was found.

In the case of a syntax error, printing the offending line would also be helpful. It is not necessary to continue scanning for additional syntax errors.

You are given a grammar for a simple calculator-style programming language. It is similar to, but different from, the example calculator language given in your text. Several input files are provided for you. Your parser will be tested against other source code in the same language.

Programming notes:

- You will need other functions besides `parse`, of course. This will be a top-down recursive-descent parser. You will need a function for each nonterminal. Of course, those functions may call sub-functions as needed. (If the subfunctions will not be needed elsewhere in the program, nest them inside the function that will use them. The principle of information hiding applies just as much in functional programming as in any other.)
- You're not limited to printing *just* a final verdict; progress messages will probably be helpful in development. Your parse tree will be a large tree structure contained in a list.
- Deliverables:
  - Submit your source code (.rkt file).
  - Submit your LLM prompts and what was returned from them if the code in them was incorporated into your program.
  - Submit a summary document about the development process, where the LLM was helpful (or not), and passing along any tips or insights you had on how to get better results.

1 There's an online book, *Beautiful Racket*, that shows how to write an interpreter for the Basic language, along with a couple of specialized languages.

Grammar, EBNF notation:

```
program -> {stmt_list} $$
stmt_list -> stmt stmt_list
stmt_list -> epsilon
stmt -> id = expr;
       | if (expr) stmt_list endif;
       | read id;
       | write expr;
expr -> id etail
       | num etail
etail -> + expr
       | - expr
        | epsilon
id -> [a-zA-Z]+
num -> numsign digit digit*
numsign -> + | - | epsilon
```