1 Lexing and Parsing

1.1 Running the Parser

Recall that there were two issues.

- Wrong Precedence

  ```
  $ evalString [] "2 * 5 + 5"
  20
  ```

  This is invalid, and is due to the fact that our grammar is **ambiguous** – there are multiple ways to parse the string `2 * 5 + 5`; one way is correct (`(2 * 5) + 5`) and one way is incorrect (`2 * (5 + 5)`). Thus, we want to tell `happy` that `*` has higher precedence than `+`.

- Wrong Associativity

  ```
  $ evalString [] "2 - 1 - 1"
  2
  ```

  So, we also need to tell `happy` that `-` is **left-associative**.

Therefore, we need to tell `happy` about precedence and associativity.

1.1.1 Solution 1: Grammar Factoring

We can split the `AExpr` non-terminal into multiple "levels."

```
Aexpr : Aexpr '+' Aexpr
      | Aexpr '-' Aexpr
      | Aexpr2

Aexpr2 : Aexpr2 '*' Aexpr2
       | Aexpr2 '/' Aexpr2
       | Aexpr3

Aexpr3 : TNUM
       | ID
       | '(' Aexpr ')'
```

Note that `AExpr` is the most general term. Then, `AExpr2` is slightly more specific; it only has expressions relating to multiplication and division, and numbers/identifiers. `AExpr3` is the most strict, with only numbers/identifiers. Intuitively, `AExpr2` will "bind tighter" than `AExpr`. We also have `AExpr3`, which is the "tightest."

This fixes the issue with `2 * 5 + 5`, but fails to parse `5 + 5`.

(Quiz.) With this new grammar, can we parse `2 - 1 - 1` the wrong way?

(a) Yes.

(b) No.

The answer is **B**. There are still multiple ways to parse `2 - 1 - 1`.

How do we fix this? One way to do so is to disallow the right-hand side of a minus to be a minus.

```
Aexpr : Aexpr '+' Aexpr2
      | Aexpr '-' Aexpr2
      | Aexpr2

Aexpr2 : Aexpr2 '*' Aexpr3
       | Aexpr2 '/' Aexpr3
       | Aexpr3

Aexpr3 : TNUM
       | ID
       | '(' Aexpr ')'
```

1.1.2 Solution 2: Parser Directives

We can just use a special syntax for the parser generator; in `happy`, this is the syntax[1]:

```
%left '+' '-'
%left '*' '/'
```

This means that

- All our operations are left-associative.
- Operators on the lower line have higher precedence.

Note that operations like *applications*, which is left-associative, do not have an operator; thus, you still need to worry about things like these.

[1] Other parser generators may use other syntax.