Compiler Project 1
Max Lapides

Error Test Cases

Test 1:

```
{
	int a
	a + 3
} $
```

Parsing error -> ERROR: expected EQUAL, found + (token 5)
This error occurs when the compiler attempts to parse `a + 3`. This is not a valid Statement as defined in the grammar.

Test 2:

```
{
	int a
	char b
	b = 3
	{
		c = "hi!"
		P(c)
	}
	c = b
	c = 4 + c
} $
```

Lexing error -> ERROR: "!" is not valid in a CharExpr (line 6, char 10)
This error occurs when the compiler attempts to lex the exclamation point character, which is not ever valid in the language.

Test 3:

```
{
	int a
	a = 2
	{
		int b
		b = 3
	P(2 + a)
	P(b)
} $
```

Parsing error -> ERROR: expected Statement, found EOF (token 24)
This program has an issue where there is a nested block that is never closed. The parser expects all blocks to be closed. Until a block is closed, the parser expects there to be another Statement.

Test 4:

```
{
	int i
	char c
	{
		i = 2
		c = 3
	}
	P(int x)
} $
```

Parsing error -> ERROR: expected Expr, found int (token 16)
Since the language only allows you to print Exprs and `int x` is a VarDecl (not an Expr), we reach an error when it tries to print `int x`.

Other errors that will be caught by the compiler include:

- Failing to end a program with a $
- Using an operator anywhere except in an IntExpr
- Including anything besides a Char between quotation marks
- Using any character not in the language (ex: ?, #, ^, etc.)
- Failing to close a CharExpr with a quotation mark
- Trying to use anything besides `int` and `char` as keywords

Successful Test Cases

Test 5:

```
int a $
```

This is an example of the simplest type of program allowed.

Test 6:

```
{
	int a
	char b
	b = 3
	{
		c = "hi"
```

```
            P(c)
            { { x = 4 } }
      }
      c = b
      c = 4 + c
      char c
} $
```

This program includes many complications. It includes nested blocks, variable declarations, variable assignments, and variable re-assignments. Note the line `c = 4 + c`. This line is valid in the language, but the seemingly equivalent `c = c + 4` is not valid in the language.

Test 7:

```
{
      int i
      char c
      {
            i = 1
            c = "xyz"
      }
      { { { } } }
      P(i)
      P(c)
      P("done")
} $
```

This program uses `i` and `c` as variable names, which is important to test because the compiler needs to distinguish these from the keywords `int` and `char`. This program also tests empty blocks and prints a CharExpr.

Test 8:

```
{
      char x
      x = "woot"
      char y
      y = x
      P(y)
      int z
      z = 2 + "two"
} $
```

This program tests assigning one variable to equal another variable, which is valid in the language. It also tests adding a digit to a CharExpr, which is also valid in the language.