Test 1:
Input:

```
{}$
{{{{{}}}}}$
{{{{}}}}}$
{int @}$
```

Output:

```
token: token_lbrace value: { at line 1

token: token_rbrace value: } at line 1

token: token_eof value: $

Lex completed for program 1 with 0 errors

--------------------------------------------------------------------------------

token: token_lbrace value: { at line 2

token: token_lbrace value: { at line 2

token: token_lbrace value: { at line 2

token: token_lbrace value: { at line 2

token: token_lbrace value: { at line 2

token: token_rbrace value: } at line 2

token: token_rbrace value: } at line 2

token: token_rbrace value: } at line 2

token: token_rbrace value: } at line 2

token: token_rbrace value: } at line 2

token: token_eof value: $

Lex completed for program 2 with 0 errors

--------------------------------------------------------------------------------

token: token_lbrace value: { at line 3

token: token_lbrace value: { at line 3

token: token_lbrace value: { at line 3

token: token_lbrace value: { at line 3

token: token_rbrace value: } at line 3

token: token_rbrace value: } at line 3

token: token_rbrace value: } at line 3

token: token_rbrace value: } at line 3

token: token_rbrace value: } at line 3

token: token_eof value: $

Lex completed for program 3 with 0 errors

--------------------------------------------------------------------------------
```

```
token: token_lbrace value: { at line 4

token: token_int value: int at line 4

Invalid Lexeme: @ at line 4

token: token_rbrace value: } at line 4

token: token_eof value: $

Lex completed for program 4 with 1 errors
```

Test 2
Input:

```
inta a1 != == string b
(}})("if"
```

Output:

```
Invalid Lexeme: inta at line 1

Invalid Lexeme: a1 at line 1

token: token_notequals value: != at line 1

token: token_doubleEquals value: == at line 1

token: token_string value: string at line 1

token: token_id value: b at line 1

token: token_lparen value: ( at line 2

token: token_rbrace value: } at line 2

token: token_rbrace value: } at line 2

token: token_rparen value: ) at line 2

token: token_lparen value: ( at line 2

token: token_quote value: " at line 2

token: token_if value: if at line 2

token: token_quote value: " at line 2

End of file token not found, but lex still completed with 2 errors
```

Test 3

Input:

```
{

  print() boolean a

)$true false
```

Output:

```
token: token_lbrace value: { at line 1

token: token_print value: print at line 2
token: token_lparen value: ( at line 2

token: token_rparen value: ) at line 2

token: token_boolean value: boolean at line 2

token: token_id value: a at line 2

token: token_rparen value: ) at line 3

token: token_eof value: $

Lex completed for program 1 with 0 errors

--------------------------------------------------------------------------------

token: token_true value: true at line 3

token: token_false value: false at line 3

End of file token not found, but lex still completed with 0 errors
```

Notes and Improvements

Testing went pretty well I think, I threw a lot at the lexer. Hopefully it can take whatever you throw at it. Before I start working on the next part of the compiler I probably should split up the findTokens() function into multiple functions because it is one ugly function! I also need to add more comments before I forget what the hell is going on. The site looks like shit, I need to clean that up as well in the next few days