1. Error Productions

Two factors contributed to the creation of error productions within the program three parser. As is pointed out in both the assignment and and the *Flex & Bison* book, the choice is between a coarse parser which may confuse errors and productions, and a higher level system of feedback which may miss errors but lead to clearer output. The productions made use of in this implementation of the parser fall on the coarse end of the spectrum due to the following reasoning presented by Levine in the aformentioned book:

> On today's computers, the interval [of execution] is more likely to be seconds, so rather than trying to guess the programmer's intentions and continue after severe errors, it makes more sense to recover as quickly as possible to a state where the programmer can revise the input and rerun the compiler.

Furthermore, as the language does not yet support block statements or a similar structure other than through line based organization, it is premature to attempt to support the block level errors structures such as loops or conditionals would introduce to the parser.

This granular approach ensures that the correct line is identified by using the semicolon as a synchronizing token, but avoid further interpretation on the input stream to avoid pointing to a potential problem that may be incorrectly diagnosed. As higher level control structures are introduced to the language, logical errors can be built on this foundation by seeing a line-based error as an atomic element in this error scheme. Thus, these errors may point to larger, structural issues while pointing the programmer to individual lines with the goal of clarifying their intentions rather than guessing at their motivation for a specific line.