Wyatt Emery Error documentation.

I have quite a few error productions in my program4.y file. I feel it is fairly comprehensive. However, I was unable to catch everything. For example,

```
class blockstuff{
  id(){
        int vardec;
        x = y;
//missing closing '}'
  id(){}
 }
```

Return syntax error and my program quits. That is the only one I am aware of that does that. I have error productions for out of order declarations, most missing braces, parentheses, semicolons and even some absent nonterminals, like block and expression. New expression was hard to get very many meaningful errors out of.

For simple errors, the line and column are pretty accurate. However, the more complicated the error, the less accurate those become. Unless, the user does some very strange things, like putting new lines in odd and unconventional places, the line number should be correct. Otherwise it will be pretty close. Sometimes the error messages are not all the helpful or just plain misleading. However, They do point you to where the error occurred and give a usually helpful message.

For any particular error production, I added it either to prevent a syntax error and quit or a "catch-all" error that says no idea, discards the token and moves on. I make sure to free discarded symbols with the %destructor command in bison.

I payed special attention to expression and statement errors because I figured they would be the most common and need to be the most comprehensive. They also have the largest number of possibilities and so provide more opportunity for error productions.

In many places, I use yylval.token to get the line and column number. This can only report the last symbol given to bison by flex. In these cases the error line and column will point to the very end of the error. However, in some cases to provide more accurate error reporting you are directed to just before the error occurred. My "catch-all" error at the top is the most generic error message, informing you I have no idea what happened, but I do know where. This discards the error token and usually allows for bison to resync at the very least, the next class declaration. Really only the first of these errors is useful. After, that it is generally just error propogation.