ChemLAB
Final Report
COMS W4115 - Programming Languages & Translators
Professor Stephen Edwards

Alice Chang (avc2120)   Gabriel Lu (ggl2110)   Martin Ong (mo2454)

December 17, 2014

Contents

Introduction

ChemLab is a language that will allow users to conveniently manipulate chemical elements. It can be used to solve chemistry and organic chemistry problems including, but not limited to, stoichiometeric calculations, oxidation-reduction reactions, acid-base reactions, gas stoichiometry, chemical equilibrium, thermodynamics, stereochemistry, and electrochemistry. It may also be used for intensive study of a molecule's properties such as chirality or aromaticity. These questions are mostly procedural and there is a general approach to solving each specific type of problem. For example, to determine the molecular formula of a compound: 1) use the mass percents and molar mass to determine the mass of each element present in 1 mole of compound 2) determine the number of moles of each element present in 1 mole of compound. Albeit these problems can generally be distilled down to a series of plug-and-chug math calculations, these calculations can become extremely tedious to work out by hand as molecules and compounds become more complex (imagine having to balance a chemical equation with Botox: $C_{6760}H_{10447}N_{1743}O_{2010}S_{32}$). Our language can be used to easily create programs to solve such problems through the use of our specially designed data types and utilities.

Chapter 1

Language Tutorial

1.1 Program Execution

`make` creates an executable `chemlab` To compile and run a `.chem` program, simply run the executable `chemlab` with your `.chem` file as the only argument. `./chemlab <program name>.chem`

It then compiles the ChemLab file into Java bytecode, which is then executed on a Java virtual machine.

1.2 Variables

Variables in ChemLAB must be declared as a specific type. To use a variable, declare the type of the variable, and assign it to the value that you want like this:

```
int myNum = 5;
String hello = "World";
```

1.3 Control Flow

ChemLAB supports *if/else* statements:

```
if(10>6){
print("inside the if");
```

```
else{
print("inside the else");
}
```

ChemLAB supports *while loops*:

```
while(i > 0){
print(i);
i = i-1;
}
```

1.4 Functions

Functions are the basis of ChemLAB. All programs in ChemLAB must contain one "main" function which is the starting point for the program. Functions can be passed any amount of parameters and are declared using the function keyword. The parameters within a function declaration must have type specifications.

This is a function that takes in two parameters:

```
function main(int A, int B){
 print A;
}
```

This is a function that takes in no parameters:

```
function main(){
print "Hello World";
}
```

1.5 Printing to stdout

To print to stdout, simply use the built-in function *print*

```
print(6);
print("Hello World");
```

Chapter 2

Language Reference Manual

2.1 Types

2.1.1 Primitive Types

There are four primitive types in ChemLab: boolean, int, double, and string.

Boolean
The boolean data type has only two possible values: true and false. The boolean data type can be manipulated in boolean expressions involving the AND, OR, and NOT operators.

Integers
Much like in the Java programming language, the int data type is represented with 32-bits and in signed two's complement form. It has a minimum value of -2^{31} and maximum value of 2^{31}. There is no automatic type conversion between a variable of type int and of type double. In fact, an error will occur when the two primitive types are intermixed.

Double
Much like in the Java programming language, a double is a double-precision 64-bit IEEE 754 floating point with values ranging from $4.94065645841246544e - 324d$ to $1.79769313486231570e + 308d$ (positive or negative). Double should be used under any circumstance when there are decimal values.

String
Unlike in the C programming language, a string is a primitive type rather than a collection of characters. A string is a sequence of characters surrounded by double

quotes “”. Our language supports string concatenation. In the context of strings, the “+” operator concatenates two strings together to form a new string.

2.1.2 Non-Primitive Types

The language comes built-in with lists, elements, molecules, equation.

Lists

A list is a collection of items that maintains the order in which the items were added much like an ArrayList in Java. The type of items in a list must be declared and the type must remain consistent throughout the lifetime of the program. A list is declared in a syntax very similar to declaration in Java:

```
<type> <identifier>[] = [ element_1, element_2, ......., element_n]
```

Element

Since there are only 118 elements, it could have been possible to hard code each element into the language. However, we chose not to do this to give the user a greater degree of flexibility in terms of declaring the properties of the element they want to consider because isotopes of elements have different amounts of neutrons and some elements can exist in more than one state. Element is declared with (atomic number, mass number, charge). The element type is the basic building block provided by the program that can be used to create molecules, compounds, etc. Elements are immutable.

${}^{12}_{6}C$ is represented as: `element C(6, 12, 0);`

${}^{14}_{6}C$ is represented as: `element C(6, 14, 0);`

Molecule

For the purpose of the language, there is no distinction between molecule or compound and both are declared the same way. A molecule is declared as a list of elements surrounded by braces.

$NaCl$ is represented as: `molecule NaCl {[Na, Cl]}`

Equation

Equation is declared in the following way: (list of elements/molecules on left side of reaction, list of elements/molecules on right side of reaction). Underneath, it is essentially, two lists that keep track of the two sides of the equation.

`<equationName>.right` or `<equationName>.left` allows easy access to one side of the equation. Once declared, an equation is immutable.

$NaOH + HCl \rightarrow NaCl + H_20$ is represented as:

```
equation NaClReaction = {[NaOH, HCl], [NaCl, H2O]};
```

2.1.3 Type Inference

The language is not type-inferred, making it necessary to explicitly declare types.

2.2 Lexical Conventions

2.2.1 Identifiers

An identifier is a sequence of letters or digits in which the first character must be a lowercase letter. Our language is case sensitive, so upper and lower case letters are considered different.

2.2.2 Keywords

The following identifiers start with a lowercase letter and are reserved for use as keywords, and may not be used otherwise:

- int
- double
- string
- boolean
- element
- molecule
- equation
- if
- else
- while
- function
- return
- true
- false
- print
- call

2.2.3 Literals

Literals are values written in conventional form whose value is obvious. Unlike variables, literals do not change in value. An integer or double literal is a sequence of digits. A boolean literal has two possible values: true or false.

2.2.4 Punctuation

These following characters have their own syntactic and semantic significance and are not considered operators or identifiers.

Punctuator	Use	Example
`,`	List separator, function parameters	`function int sum(int a, int b);`
`;`	Statement end	`int x = 3;`
`"`	String declaration	`string x = "hello";`
`[]`	List delimiter	`int x[] = [1, 2, 3];`
{}	Statement list deliminiter, and element/molecule/equation declaration	`if(expr)` { `statements` }
`()`	Conditional parameter delimiter, expression precedence	`while(i > 2)`

2.2.5 Comments

Much like in the C programming language, the characters `/*` introduce a comment, which terminates with the characters `*/`. Single line comments start with `//` and end at the new line character \`n`.

2.2.6 Operators

Operator	Use	Associativity
`=`	Assignment	Right
`==`	Test equivalence	Left
`!=`	Test inequality	Left
`>`	Greater than	Left
`<`	Less than	Left
`>=`	Greater than or equal to	Left
`<=`	Less than or equal to	Left
`&&`	AND	Left
`||`	OR	Left
`.`	Access	Left
`*`	Multiplication	Left
`/`	Division	Left
`+`	Addition	Left
`-`	Subtraction	Left
`^`	Concatenate	Left
`%`	Modulo	Left

The precedence of operators is as follows (from highest to lowest):

1. `* / %`
2. `+ -`
3. `< > <= >=`
4. `== !=`
5. `&&`
6. `||`
7. `.`
8. `^`
9. `=`

2.3 Syntax

A program in ChemLab consists of at least one function, where one of them is named "main". Within each function there is a sequence of zero or more valid ChemLab statements.

2.3.1 Expressions

An expression is a sequence of operators and operands that produce a value. Expressions have a type and a value and the operands of expressions must have compatible types. The order of evaluation of subexpressions depends on the precedence of the operators but, the subexpressions themselves are evaluated from left to right.

Constants

Constants can either be of type boolean, string, int, or double.

Identifiers

An identifier can identify a primitive type, non-primitive type, or a function. The type and value of the identifier is determined by its designation. The value of the identifier can change throughout the program, but the value that it can take on is restricted by the type of the identifier. Furthermore, after an identifier is declared, there can be no other identifiers of the same name declared within the scope of the whole program.

```
int x = 3;
x = true; //syntax error
boolean x = 5; //error, x has already been declared
```

Binary Operators

Binary operators can be used in combination with variables and constants in order to create complex expressions. A binary operator is of the form : `<expression> <binary-operator> <expression>`

Arithmetic operators Arithmetic operators include `*`, `/`, `%`, `+`, and `-`. The operands to an arithmetic operator must be numbers. the type of an arithmetic operator expression is either an int or a double and the value is the result of calculating the expression. Note, can not do arithmetic operations when the values involved are a mix of int and double.

`expression * expression`
The binary operator `*` indicates multiplication. It must be performed between two int types or two double types. No other combinations are allowed.

`expression / expression`
: The binary operator `/` indicates division. The same type considerations as for multiplication apply.

`expression % expression`
: The binary operator `%` returns the remainder when the first expression is divided by the second expression. Modulo is only defined for int values that have a positive value.

`expression + expression`
: The binary operator `+` indicates addition and returns the sum of the two expressions. The same type considerations as for multiplication apply.

`expression - expression`
: The binary operator `-` indicates subtraction and returns the difference of the two expressions. The same type considerations as for multiplication apply.

Relational operators Relational operators include `<`, `>`, `<=`, `>=`, `==`, and `!=`. The type of a relational operator expression is a boolean and the value is true if the relation is true while it is false if the relation is false.

`expression1 > expression2`
: The overall expression returns true if expression1 is greater than expression 2

`expression1 < expression2`
: The overall expression returns true if expression1 is less than expression 2

`expression1 >= expression2`
: The overall expression returns true if expression1 is greater than or equal to expression 2

`expression1 <= expression2`
: The overall expression returns true if expression1 is less than or equal to expression 2

`expression1 == expression2`
: The overall expression returns true if expression1 is equal to expression 2.

`expression1 != expression2`
: The overall expression returns true if expression1 is not equal to expression 2

Assignment operator The assignment operator (`=`) assigns whatever is on the right side of the operator to whatever is on the left side of the operator

`expression1 = expression2`
: expression1 now contains the value of expression2

Access operator The access operator is of the form expression.value. The expression returns the value associated with the particular parameter. The expression must be of a non-primitive type.

Logical operators Logical operators include AND (`&&`) and OR (`||`). The operands to a logical operator must both be booleans and the result of the expression is also a boolean.

`expression1 && expression2`
The overall expression returns true if and only if expression1 evaluates to true and expression2 also evaluates to true.

`expression1 || expression2`
The overall expression returns true as long as expression1 and expression2 both do not evaluate to false.

Parenthesized Expression

Any expression surrounded by parentheses has the same type and value as it would without parentheses. The parentheses merely change the precedence in which operators are performed in the expression.

Function Creation

The syntax for declaration of a function is as follows

```
function functionName (type parameter1, type parameter 2, ...) {
 statements
}
```

The function keyword signifies that the expression is a function. Parameter declaration is surrounded by parentheses where the individual parameters are separated by commas. All statements in the function must be contained within the curly braces. A good programming practice in ChemLab is to declare all the functions at the beginning of the program so that the functions will definitely be recognized within the main of the program.

Function Call

Calling a function executes the function and blocks program execution until the function is completed. When a function is called, the types of the parameter passed into the function must be the same as those in the function declaration. The way to call a function is as follows using the Call keyword: call functionName(param1, param2, etc...) When a function with parameters is called, the parameters passed into the function are evaluated from left to right and copied by value into the function's scope. functionName( ) if there are no parameters for the function

2.3.2 Statements

A statement in ChemLab does not produce a value and it does not have a type. An expression is not a valid statement in ChemLab.

Selection Statements

A selection statement executes a set of statements based on the value of a specific expression. In ChemLab, the main type of selection statement is the if-else statement. An if-else statement has the following syntax:

```
 if( expression){

 }else{

 }
```

Expression must evaluate to a value of type boolean. If the expression evaluates to true, then the statements within the first set of curly brackets is evaluated. If the expression evaluates to false, then the statements in the curly brackets following else is evaluated. If-else statements can be embedded within each other. Much like in the C programming language, the dangling if-else problem is resolved by assigning the else to the most recent else-less if. Unlike in Java, an if must be followed by an else. A statement with only if is not syntactically correct.

```
if ( ){
 if ( ){

 }else{

 }
}else{

}
```

Iteration Statements

ChemLab does not have a for loop unlike most programming languages. The only iteration statement is the while loop. The while statements evaluates an expression before going into the body of the loop. The expression must be of type boolean and the while loop while continue executing so long as the expression evaluates to true. Once the expression evaluates to false, the while loop terminates. The while loop syntax is as follows:

```
while ( expression ) {
 statements
}
```

Note that if values in the expression being evaluated are not altered through each iteration of the loop, there is a risk of going into an infinite loop.

Return Statements

A return statement is specified with the keyword return. In a function, the expression that is returned must be of the type that the function has declared. The syntax of a return statement is: return expression;

The return statement will terminate the function it is embedded in or will end the entire program if it is not contained within a function.

2.3.3 Scope

A block is a set of statements that get enclosed by braces. An identifier appearing within a block is only visible within that block. However, if there are two nested blocks, an identifier is recognizable and can be edited within the nested block.

```
function int notRealMethod(int x){
int y = 4;
while(x>5){
 while(z>2){
 y++;
 }
}
```

In this case, y is recognizable within the second while loop and its value will be incremented. One must also note that, functions only have access to those identifiers that are either declared within their body or are passed in as parameters.

2.4 Built-in Functions

Balance Equations
Given an unbalanced equation, this utility will be able to compute the correct coefficients that go in front of each molecule to make it balanced. The balance function only takes molecule types.

Molar Mass Calculation
Given a molecule, this utility will be able to compute the total molar mass of the molecule

Naming of Molecules
Given a molecule, the utility will print out the name in correct scientific notation (ex. $H_2 0$ will be printed as Dihydrogen Monoxide)

Printing of Equations
Given an equation, the utility will print out the equation in correct scientific notation

Amount of Moles
Given the element and the amount of grams of the element, this utility will return the amount of moles of the element.

Chapter 3

Project Plan

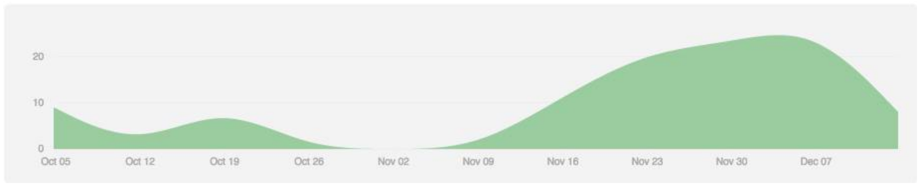
Like any project, careful planning and organization is paramount to the success of the project. More importantly however, is the methodical execution of the plan. Although we originally developed a roadmap for success as well as implemented a number of project management systems, we did not follow the plan as intended. This section outlines our proposed plans for making ChemLAB happen and the actual process that we went through.

3.1 Proposed Plan

We had originally planned to use the waterfall model in our software development process in which we would first develop a design for our language, followed by implementation, and finally testing. The idea was for all team members to dedicate complete focus to each stage in the project. Especially since we only had three members on our team, our roles were not as distinct and everyone had the chance to work, at least in some capacity, in all the roles. We intended to meet consistently each week on for at least two hours. During our meetings, each member was suppose to give an update about what he or she had been working on the past week as well as plans for the upcoming week and any challenges he or she faced that required the attention of the rest of the group. To help facilitate communication and the planning of meetings, we used Doodle to vote on what times were best for meetings. Also, in order to improve team dynamics, we planned to meet at least once every two weeks outside the context of school in order to hang out and have fun. Development would occur mostly on Mac OS and Windows 7, using the latest versions of OCaml, Ocamllex, and OCamlyacc for the compiler. We used Github for version control and makefiles to ease the work of compiling and testing code. The project timeline that we had laid out at the beginning was as follows:

- Sept 24th: Proposal Due Date
- Oct 2nd: ChemLAB syntax roughly decided upon
- Oct 23th: Scanner/Parser/AST unambiguous and working
- Oct 27th: LRM Due Date
- Nov 9th: Architectural design finalized
- Dec 5th: Compile works, all tests passed
- Dec 12th: Project report and slides completed
- Dec 17th: Final Project Due Date

3.2 What Actually Happened



This graph was pulled from Github reflecting the number of commits being made over the span of this semester. Due to schedule conflicts and a false sense of security, we did not start intensely working on the project until after Thanksgiving break. Since we did not coordinate the development of the Scanner, AST, and parser with the writing of the LRM, our language did not have as concrete a structure as we had hoped. Furthermore, we did not have enough time to implement some of the features in our language such as object-orientation or more built-in functions. As we were developing the software, we did make sure to allow testing at all steps in the design process. In the test script, we had identifiers for how far in the compilation process we wanted the program to run. Thus, we were able to maintain testing capabilities even before all of our code was ready. We discuss the testing procedure in more detail in a subsequent section.

3.3 Team Responsibilities

This subsection describes the contributions made by each team member:

Project Proposal Gabriel L/Alice C/Martin O

Scanner	Gabriel L
AST	Alice C/Gabriel L/Martin O
Parser	Alice C/Martin O
LRM	Gabriel L
Code Generation	Alice C
Top-Level ChemLAB Executable	Martin O
Semantic Analyzer	Gabriel L/Martin O
Testing	Martin O
Final Report	Gabriel L/Martin O

3.4 Project Log

See Appendix C.

Chapter 4

Architectural Design

The architectural design of ChemLAB can be divided into the following steps

1. Scanning
2. Parsing
3. Semantic Analysis
4. Java code generation
5. Running the Java code

4.1 Scanning

The ChemLAB scanner tokenizes the input into ChemLAB readable units. This process involves discarding whitespaces and comments. At this stage, illegal character combinations are caught. The scanner was written with ocamllex.

4.2 Parsing and Abstract Syntax Tree

The parser generates an abstract syntax tree based on the tokens that were provided by the scanner. Any syntax errors are caught here. The parser was written with ocamlyacc.

4.3 Semantic Analysis

The semantic analyzer takes in the AST that was generated by the parser and checks the AST for type errors as well as to make sure that statements and expressions are written in a way that corresponds to the syntax defined by the language. A semantically checked AST (SAST) is not generated. If no errors are thrown, then we can assume that it is safe to use the AST to generate Java code.

4.4 Java Generation

The module walks the AST and generates Java code corresponding to the program. All of the code is put into two Java files. One contains graphics and one contains everything else related to the program. The Java code is generated but not compiled. This needs to be done by the ChemLAB script which will run the javac command.

Chapter 5

Test Plan

5.1 Introduction

To ensure that one person's change and updates would not affect the changes others made previously, an automated test was put in place to run through all the tests to make sure everything that worked before still continued to work. Testing was done using a bash shell script to automate the process. The shell script compiles and runs all the test files and compares them with the expected output. Test cases were written to test individual components of the language such as arithmetic, conditional loops, printing, etc.

5.2 Sample Test Cases

See Appendix B.

Chapter 6

Lessons Learned

6.1 Alice Chang

“Never have I spent so much time on so little code that does so much” adequately sums up my experience this semester in Edwards Programming Languages and Translators class. Indeed, it was a perpetual struggle at first to get the hang of OCaml, which was like no other language I had tackled in the past. Yeah sure, it was essentially java, but upside down and insides out. Initially I entered this class with little knowledge of how a parser or compiler worked. Composing a project proposal knowing so little felt like a clumsy and fruitless attempt to fly when we barely knew how to walk. Yet throughout the course of the semester, Ive gained much more than knowledge to build a compiler but also skills to work in a team and most importantly the ability to reassure the heart at times of desperation that everything was going to be all right despite the rapidly approaching deadline.

Our team was one man (or woman) short as we had three members. Despite of the slight disadvantage, we learned to view it as a mixed blessing as it was easy to find time to meet up. However as we soon learned, three heads was not always better than one, only when put together did we slowly start to compose our compiler. We experimented with multiple ways of programming: The Lonely All-Nighter in which we all stayed up coding separate codes that worked individually but would not compile as a whole and eventually The Cozy Campfire solution in which one person was primarily in charge of coding and two people gathered around providing feedback. Yet the Cozy Campfire also had its downfalls too lots of ideas being expressed simultaneously and very little progress. Essentially it was like two overly opinionated backseat drivers bickering back and forth while the driver sat in baffled silence. We had so many ideas going on at once that it was often difficult for the programmer to follow so eventually we broke down our ideas into small milestones and accomplished them through a step-by-step procedure.

As we near our presentation date, weve gotten closer as a team and learned to manage our time well, communicate with teammates, and decipher cryptic existing code. Like soldiers in combat, our team suffered through endless out-of-bounds errors and bonded through several panic attacks when GitHub repeated crashed on us. Yet at the end of this class well have earned our wings to soar through parsers, interpreters, and compilers and wear with us these experiences like a badge noble achievementthat is, at least until next semester when we take another class that will once again challenge our late-night coding abilities. Yet undoubtedly, the lessons learned through this semester will stay with us beyond this class.

6.2 Gabriel Lu

I have learned that writing code is not the only part of software development. I think the teamwork and planning is even more important to the success of the project than having superstar coders. Without this solid foundation and the ability to communicate effectively, a team can be extremely handicapped from the beginning. Since this was my first experience working in a team to develop software, I definitely made mistakes in my capacity as the project manager. I learned that the PM has to take more initiative to enforce soft deadlines as well as to ensure that meetings are productive. I also learned the difficulty in being a good PM from the aspect of being able to motivate your teammates in a constructive manner. We ran into difficulties with Github in consolidating changes throughout the course of the project and also in being productive during meetings. An interesting solution to the lack of productivity during meetings was paired programming which I learned could be a good way to develop code. I learned the importance of team chemistry and the importance of working on a project that everyone has a passion for and invested interest in seeing come to fruition. Even though it was a struggle the last two weeks to start and finish the project, I was glad that I was working on the project with two good friends. I think that if we had chosen a language that we all were more passionate about and had better planning, we could have created an even better product. Overall though, I learned a lot about working in a team, OCaml, and the difficulty in developing a language.

6.3 Martin Ong

The most important lesson I learned from this project is that communication between members of the group is paramount and GitHub can be your friend. Often times, the most difficult problem we encountered was trying to understand the code other members have written and be able to incorporate their code in our own work. At times, the lack of communication led to clashes in our work where a person would change code back to what they thought was working, when in fact they were undoing the work of another person. This was also due to our unfamiliarity with GitHub. Before this class, most of us had only used GitHub for individual projects, so when conflicts came up and we had to merge them, often times the response was to freak out. Resolving these conflicts on GitHub were not easy as changes another person made looked like it didn't belong there to the person resolving the conflict. The mantra then was to "just make it work", so sometimes progress another person made was disappeared in this way.

If we could do this over again, I would definitely split up the project clearly into concrete slices for each member to take ownership on, such as having one person be in charge of one file. This way one person could keep track of everything that still needs to be done for a particular file. We worked in a non-hierarchical way where we would meetup and code together on the same computer. This led to a decrease in productivity, even though everyone could understand what the code did in the end. Having one less person in our group also put us at a disadvantage, because, even though it made it easier to schedule meetings with each other, each of us had to do much more.

I would also create small milestone deadlines to complete throughout the semester to be more efficient. Since this is probably one of the largest coding projects we have ever done, it did not hit us until Thanksgiving break that there were much more than we had anticipated. I believe that if we put more effort in the beginning to get a good foundation for the ast, scanner and parser, it would be easier working with the other components.

I must say, I have learned a lot about working as a team under a coding environment. We have definitely learned and changed a lot through this project, both in terms of OCaml and working as a team.

Appendix A

Code Listing

Listing A.1: Abstract Syntax Tree (ast.ml)

```
type operator = Add | Sub | Mul | Div | Mod
type rop = Eq | Neq | Lt | Leq | Gt | Geq
type re = And | Or
type bool = True | False
type data_type = IntType | BooleanType | StringType | DoubleType |
    ElementType | MoleculeType | EquationType
type element = Element of string
type molecule = Molecule of string

type expr =
    Binop of expr * operator * expr
  | Brela of expr * re * expr
  | Int of int
  | String of string
  | Boolean of expr * rop * expr
  | Double of float
  | Asn of string * expr
  | Equation of string * element list * element list
  | Balance of molecule list * molecule list
  | Concat of expr * expr
  | Print of expr
  | List of expr list
  | Call of string * expr list
  | Access of expr * string
  | Bracket of expr
  | Charge of string
  | Electrons of string
  | Mass of string
  | Null
  | Noexpr

```

```
type stmt =
    Block of stmt list
  | Expr of expr
  | Return of expr
  | If of expr * stmt * stmt
  | For of expr * expr * expr * stmt
  | While of expr * stmt
  | Print of expr
  | Draw of string * int * int * int * int * int * int * int * int

type variable_decl = {
  vname : string;
  vtype : data_type;
}

type element_decl = {
  name : string;
  mass : int;
  electrons : int;
  charge : int;
}

type molecule_decl = {
  mname : string;
  elements: element list;
}




type par_decl = {
  paramname : string; (* Name of the variable *)
  paramtype : data_type; (* Name of variable type *)
}

type func_decl = {
  fname : string;
  formals : par_decl list;
  locals: variable_decl list;
  elements : element_decl list;
  molecules : molecule_decl list;
  body : stmt list;
}

(* type program = {
  gdecls : var_decl list;
  fdecls : func_decl list
}
 *)
type program = func_decl list
```

Listing A.2: Scanner (`scanner.mll`)

```
{ open Parser }

let digit = ['0'-'9']
let letter = ['A'-'Z' 'a'-'z']
let element = ['A'-'Z']['a'-'z']?   (* Symbol of element such as: H, Cl *)

rule token = parse
    [' ' '\t' '\r' '\n']           { token lexbuf }
  | "/*"                       { comment lexbuf }
  | "//"                       { line_comment lexbuf }
  | '('                          { LPAREN }
  | ')'                        { RPAREN }
  | '['                        { LBRACKET }
  | ']'                        { RBRACKET }
  | '{'                        { LCURLY }
  | '}'                        { RCURLY }
  | '"'                        { STRINGDECL }
  | ';'                                { SEMI }
  | ':'                                { COLON }
  | ','                                { COMMA }
  | '.'                                { ACCESS }
  | '+'                                { PLUS }
  | '-'                                { MINUS }
  | '*'                                { TIMES }
  | '/'                                { DIVIDE }
  | '%'                                { MOD }
  | '='                                { ASSIGN }
  | '^'                                { CONCAT }
  | "=="                               { EQ }
  | "!="                               { NEQ }
  | '<'                                { LT }
  | "<="                               { LEQ }
  | '>'                                { GT }
  | ">="                               { GEQ }
  | "&&"                               { AND }
  | "||"                               { OR }
  | '!'                                { NOT }
  | "-->"                      { ARROW }
  | "if"                         { IF }
  | "else"                       { ELSE }
  | "while"                      { WHILE }
  | "for"                      { FOR }
  | "int"                        { INT }
  | "double"                       { DOUBLE }
  | "string"                     { STRING }
  | "boolean"                      { BOOLEAN }
  | "element"                      { ELEMENT }
  | "molecule"                     { MOLECULE}
  | "equation"                     { EQUATION }
  | "balance"                { BALANCE }
  | "mol_mass"                 { MASS }
  | "mol_charge"                 { CHARGE }
  | "mol_electrons"              { ELECTRONS }
  | "mass"     as attr         { ATTRIBUTE(attr) }
  | "charge"     as attr         { ATTRIBUTE(attr) }
  | "electrons" as attr          { ATTRIBUTE(attr) }
  | "function"                 { FUNCTION }
  | "object"                   { OBJECT }
  | "return"                   { RETURN }
  | "print"                  { PRINT }
  | "call"                 { CALL }
  | "draw"                 { DRAW }
  | "true"                               { BOOLEAN_LIT(true) }
  | "false"                              { BOOLEAN_LIT(false) }
  | (digit)+ '.' (digit)+ as lxm                  { DOUBLE_LIT(float_of_string
    lxm) }
  | digit+ as lxm                          { INT_LIT(int_of_string lxm) }
  | element as lxm                { ELEMENT_LIT(lxm)}
  | (element ['0'-'9']*)+ as lxm          { MOLECULE_LIT(lxm)}
  | ['a'-'z'](letter | digit | '_')* as lxm        { ID(lxm)}
  | '"' [^'"']* '"'  as lxm                    { STRING_LIT(lxm) }
  | eof                            { EOF }
  | _ as char                { raise (Failure("illegal character " ^
                          Char.escaped char)) }


and comment = parse
    "*/"           { token lexbuf }
  | _            { comment lexbuf }

and line_comment = parse
    "\n"           { token lexbuf }
  | _            { line_comment lexbuf }
```

Listing A.3: Parser (`parser.mly`)

```
%{ open Ast
  let parse_error s = (* Called by parser on error *)
    print_endline s;
    flush stdout
%}

%token SEMI LPAREN RPAREN LBRACKET RBRACKET LCURLY RCURLY COMMA STRINGDECL
    COLON ACCESS CONCAT NOT OBJECT ARROW
%token PLUS MINUS TIMES DIVIDE MOD PRINT ASSIGN
%token EQ NEQ LT LEQ GT GEQ EQUAL
%token RETURN IF ELSE FOR WHILE INT DOUBLE STRING BOOLEAN ELEMENT MOLECULE
    EQUATION FUNCTION
%token INT DOUBLE STRING BOOLEAN ELEMENT MOLECULE EQUATION FUNCTION
```

```
%token CALL ACCESS DRAW
%token BALANCE MASS CHARGE ELECTRONS
%token AND OR
%token INT BOOLEAN STRING DOUBLE
%token <string> DATATYPE ATTRIBUTE
%token <bool> BOOLEAN_LIT
%token <string> ELEMENT_LIT
%token <string> MOLECULE_LIT
%token <string> STRING_LIT
%token <string> ID
%token <int> INT_LIT
%token <float> DOUBLE_LIT
%token EOF


%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left CONCAT
%left ACCESS
%left OR
%left AND
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE MOD
%nonassoc LPAREN RPAREN

%start program
%type <Ast.program> program

%%
program:
  /* nothing */        { [] }
  | program fdecl        { ($2 :: $1) }

id:
    ID              { $1 }
  | STRING_LIT         { $1 }
  | ELEMENT_LIT          { $1 }
  | MOLECULE_LIT         { $1 }

element:
  ELEMENT_LIT               { Element($1) }

molecule:
    MOLECULE_LIT              { Molecule($1) }

vdecl:
  datatype ID SEMI
  { { vname = $2;
    vtype = $1;
  } }

vdecl_list:
  /* nothing */    {[]}
  | vdecl_list vdecl  {($2::$1)}

stmt:
    expr SEMI                            { Expr($1) }
  | RETURN expr SEMI                        { Return($2) }
  | PRINT expr SEMI                         { Print($2) }

  | DRAW LPAREN STRING_LIT COMMA INT_LIT COMMA INT_LIT COMMA INT_LIT COMMA
      INT_LIT COMMA INT_LIT COMMA INT_LIT COMMA INT_LIT COMMA INT_LIT RPAREN
      SEMI  { Draw($3, $5, $7, $9, $11, $13, $15, $17, $19) }

  | LCURLY stmt_list RCURLY                    { Block(List.rev $2) }
  | IF LPAREN expr RPAREN stmt %prec NOELSE              { If($3, $5, Block
      ([]) ) }
  | IF LPAREN expr RPAREN stmt ELSE stmt               { If($3, $5, $7) }
  | FOR LPAREN expr SEMI expr SEMI expr RPAREN stmt          { For($3, $5, $7,
      $9) }
  | WHILE LPAREN expr RPAREN stmt                     { While($3, $5) }

stmt_list:
  /* nothing */    { [] }
  | stmt_list stmt  { ($2 :: $1) }

datatype:
    INT      { IntType }
  | BOOLEAN    { BooleanType }
  | STRING   { StringType }
  | DOUBLE   { DoubleType }

expr:
    INT_LIT                              { Int($1) }
  | id                               { String($1) }
  | EQUATION id LCURLY element_list ARROW element_list RCURLY   { Equation($2
      , $4, $6) }
  | BALANCE LPAREN molecule_list ARROW molecule_list RPAREN     { Balance($3,
      $5) }
  | expr ACCESS ATTRIBUTE                       { Access($1, $3) }
  | expr PLUS expr                          { Binop($1, Add, $3) }
  | expr MINUS expr                           { Binop($1, Sub, $3) }
  | expr TIMES expr                           { Binop($1, Mul, $3) }
  | expr DIVIDE expr                          { Binop($1, Div, $3) }
  | expr MOD expr                           { Binop($1, Mod, $3) }

  | expr EQ expr                            { Boolean($1, Eq,  $3) }
  | expr NEQ expr                         { Boolean($1, Neq, $3) }
  | expr LT expr                          { Boolean($1, Lt,  $3) }
  | expr GT expr                          { Boolean($1, Gt,  $3) }
  | expr LEQ expr                         { Boolean($1, Leq, $3) }
  | expr GEQ expr                         { Boolean($1, Geq, $3) }

  | expr AND expr                         { Brela($1, And, $3) }
  | expr OR expr                          { Brela($1, Or,  $3) }
  | expr CONCAT expr                        { Concat($1, $3) }
  | id ASSIGN expr                        { Asn($1, $3) }
  | CALL id LPAREN actuals_opt RPAREN              { Call($2, $4) }
  | LPAREN expr RPAREN                      { Bracket($2) }
  | CHARGE LPAREN id RPAREN                   { Charge($3) }
  | MASS LPAREN id RPAREN                   { Mass($3) }
  | ELECTRONS LPAREN  id RPAREN                 { Electrons($3) }

edecl:
  ELEMENT id LPAREN INT_LIT COMMA INT_LIT COMMA INT_LIT RPAREN SEMI
  {{
    name = $2;
    mass = $4;
    electrons = $6;
    charge = $8
  }}

edecl_list:
  /* nothing */         { [] }
  | edecl_list edecl        { List.rev ($2 :: $1)}


element_list:
    element               { [$1] }
  | element_list COMMA element    { ($3 :: $1)}

molecule_list:
    molecule               { [$1] }
  | molecule_list COMMA molecule    { ($3 :: $1)}

mdecl:
  MOLECULE id LCURLY element_list RCURLY SEMI
  {{
    mname = $2;
    elements = $4;
  }}

mdecl_list:
  /* nothing */         { [] }
  | mdecl_list mdecl        { ($2 :: $1) }

formals_opt:
  /* nothing */           { [] }
  | formal_list           { List.rev $1 }

formal_list:
    param_decl           { [$1] }
  | formal_list COMMA param_decl  { $3 :: $1 }

actuals_opt:
    /* nothing */          { [] }
  | actuals_list           { List.rev $1 }

actuals_list:
    expr                         { [$1] }
  | actuals_list COMMA expr      { $3 :: $1 }

param_decl:
  datatype id
    { { paramname = $2;
      paramtype = $1 } }

fdecl:
   FUNCTION id LPAREN formals_opt RPAREN LCURLY vdecl_list edecl_list
       mdecl_list stmt_list RCURLY
  { {
    fname = $2;
    formals = $4;
    locals = List.rev $7;
    elements =  List.rev $8;
    molecules = List.rev $9;
    body = List.rev $10
  } }
```

Listing A.4: Semantic Checker (`semantic.ml`)

```
open Ast
open Str

type env = {
  mutable functions : func_decl list;
}

let function_equal_name name = function
  func-> func.fname = name

let function_fparam_name name = function
  par -> par.paramname = name

let function_var_name name = function
  variable -> variable.vname = name

```

```
(* Checks whether a function has been defined duplicately *)
let function_exist func env =
  let name = func.fname in
     try
       let _ = List.find (function_equal_name name) env.functions in
         let e = "Duplicate function: "^ name ^" has been defined more than
             once" in
           raise (Failure e)
    with Not_found -> false


(*Checks if function has been declared*)
let exist_function_name name env = List.exists (function_equal_name name) env
    .functions


let get_function_by_name name env =
  try
       let result = List.find (function_equal_name name) env.functions in
            result
  with Not_found -> raise(Failure("Function "^ name ^ " has not been declared
      !"))


let get_formal_by_name name func =
  try
    let result = List.find(function_fparam_name name) func.formals in
      result
  with Not_found -> raise(Failure("Formal Param" ^ name ^ " has not been
      declared!"))

let get_variable_by_name name func =
  try
    let result = List.find(function_var_name name) func.locals in
    result
  with Not_found -> raise(Failure("Local Variable " ^ name ^ "has not been
      declared!"))


let count_function_params func = function
  a -> let f count b =
  if b = a
    then count+1
    else count
in
  let count = List.fold_left f 0 func.formals in
    if count > 0
      then raise (Failure("Duplicate parameter in function " ^ func.fname))
      else count

```

```

let count_function_variables func = function
  a -> let f count b =
  if b = a
    then count+1
    else count
in
  let count = List.fold_left f 0 func.locals in
    if count > 0
      then raise (Failure("Duplicate variable in function " ^ func.fname))
      else count

(*Determines if a formal paramter with the given name   fpname   exits in
    the given function*)

let exists_formal_param func fpname =
try
 List.exists (function_fparam_name fpname) func.formals
with Not_found -> raise (Failure ("Formal Parameter " ^ fpname ^ " should
    exist but was not found in function " ^ func.fname))


(*Determines if a variable declaration with the given name   vname   exists
     in the given functioin*)

let exists_variable_decl func vname =
try
 List.exists (function_var_name vname) func.locals
with Not_found -> raise (Failure ("Variable " ^ vname ^ " should exist but
    was not found in function " ^ func.fname))


let dup_param_name func fpname =
  let name = func.formals in
    try
      List.find (function name -> name.paramname = fpname.paramname ) name
  with Not_found -> raise (Failure ("Duplicate param names"))



let get_fparam_type func fpname =
  let name = func.formals in
    try
      let fparam = List.find(function_fparam_name fpname) name in
        fparam.paramtype
    with Not_found -> raise (Failure ("Formal param should exist but not
        found"))


(*given variable name, get type*)
```

```
let get_var_type func vname =
  let name = func.locals in
    try
      let var = List.find(function_var_name vname) name in
        var.vtype
    with Not_found -> raise (Failure ("Variable should exist but not found"))

(*Determines if the given identifier exists*)
let exists_id name func = (exists_variable_decl func name) || (
    exists_formal_param func name)

(*see if there is a function with given name*)
let find_function func env =
 try
 let _ = List.find (function_equal_name func) env.functions in
 true (*return true on success*)
 with Not_found -> raise Not_found

let is_int s =
 try ignore (int_of_string s); true
 with _ -> false

let is_float s =
 try ignore (float_of_string s); true
 with _ -> false

let is_letter s = string_match (regexp "[A-Za-z]") s 0

let is_string s = string_match (regexp "\".*\"") s 0

let is_string_bool = function "true" -> true | "false" -> true | _ -> false

let rec is_num func = function
    Int(_) -> true
  | Double(_) -> true
  | Binop(e1,_,e2) -> (is_num func e1) && (is_num func e2)
  | _ -> false

let rec is_boolean func = function
  Boolean(_) -> true
  | _ -> false

let rec get_expr_type e func =
  match e with
    | String(s) -> StringType
    | Int(s) -> IntType
    | Double(f) -> DoubleType
    | Boolean(_,_,_) -> BooleanType
    | Binop(e1,_,e2) -> get_expr_type e1 func
    | Brela(e1,_,e2) -> BooleanType
    | Asn(expr, expr2) -> get_expr_type expr2 func
    | Concat(s, s2) -> StringType
    | Bracket(e1) -> get_expr_type e1 func
    | Access(id,attr) -> IntType (* Call only returns mass, charge, or electrons *)
    | Equation (_, _, _) -> EquationType
    | Balance (_, _) -> StringType
    | Print _ -> StringType
    | List _ -> IntType
    | Call (_, _) -> IntType
    | Charge _ -> IntType
    | Electrons _ -> IntType
    | Mass _ -> IntType
    | Null -> IntType
    | Noexpr -> IntType

let rec valid_expr (func : Ast.func_decl) expr env =
  match expr with
    Int(_) -> true
  | Double(_) ->  true
  | Boolean(_) -> true
  | String(_) -> true
  | Binop(e1,_,e2) -> let t1 = get_expr_type e1 func and t2 = get_expr_type e2 func in
      begin
        match t1, t2 with
          DoubleType, DoubleType -> true
        | IntType, IntType -> true
        | _,_ -> raise (Failure "Types for binary expresion must be matching int or double")
      end
  | Brela (e1,_,e2) -> let t1 = get_expr_type e1 func and t2 = get_expr_type e2 func in
      begin
        match t1, t2 with
          BooleanType, BooleanType -> true
        | _,_ -> raise (Failure "Invalid type for AND, OR expression")
      end
  | Asn(id, expr2) ->
    begin
      let t1 = get_var_type func id and t2 = get_expr_type expr2 func in
        match t1,t2 with
          StringType, StringType -> true
        | IntType, IntType -> true
        | DoubleType, DoubleType -> true
        | ElementType, ElementType -> true
        | MoleculeType, MoleculeType -> true
        | EquationType, EquationType -> true
        | IntType, StringType -> true
        | StringType, IntType -> true
      | _,_ -> raise(Failure ("DataTypes do not match up in an assignment
          expression to variable "))
    end
  | Concat(e1,e2) ->
    begin
      match get_expr_type e1 func, get_expr_type e2 func with
            StringType, StringType -> true
        | _,_ -> raise(Failure("Concatenation only works between two strings"
          ))
    end
  | Call(f_name,_) -> exist_function_name f_name env
  | List(e_list) -> let _ = List.map (fun e -> valid_expr func e env) e_list
      in true
  | _ -> true

let has_return_stmt list =
  if List.length list = 0
    then false
    else match (List.hd (List.rev list)) with
      Return(_) -> true
    | _ -> false

let has_return_stmt func =
  let stmt_list = func.body in
    if List.length stmt_list = 0
      then false
      else match List.hd (List.rev stmt_list), func.fname with
        Return(e),"main" -> raise(Failure("Return statement not permitted in
            main method"))
      | _, "main" -> false
      | Return(e), _ -> true
      | _,_ -> false


(*Returns the type of a given variable name *)
let get_type func name =
  if exists_variable_decl func name (* True if there exists a var of that
      name *)
    then get_var_type func name
    else
      if exists_formal_param func name
        then get_fparam_type func name
        else (*Variable has not been declared as it was not found*)
          let e = "Variable \"" ^ name ^ "\" is being used without being
              declared in function \"" ^ func.fname ^ "\"" in
            raise (Failure e)


(* Check that the body is valid *)
let valid_body func env =
  (* Check all statements in a block recursively, will throw error for an invalid stmt *)
  let rec check_stmt = function
      Block(stmt_list) -> let _ = List.map(fun s -> check_stmt s) stmt_list in
        true
    | Expr(expr) -> let _ = valid_expr func expr env in
      true
    | Return(expr) -> let _ = valid_expr func expr env in
      true
    | If(condition, then_stmts, else_stmts) -> let cond_type = get_expr_type condition func in
      begin
        match cond_type with
            BooleanType ->
              if (check_stmt then_stmts) && (check_stmt else_stmts)
                then true
                else raise( Failure("Invalid statements in If statement within function \"" ^ func.fname ^ "\""))
          | _ -> raise( Failure("Condition of If statement is not a valid boolean expression within function \"" ^ func.fname ^ "\"") )
      end
    | For(init, condition, do_expr, stmts) -> let cond_type = get_expr_type condition func in
      let _ = valid_expr func do_expr env in
        let _ = valid_expr func init env in
          begin
            match cond_type with
                BooleanType ->
                  if check_stmt stmts
                    then true
                    else raise( Failure("Invalid statements in For loop within function \"" ^ func.fname ^ "\""))
              | _ -> raise( Failure("Condition of For loop is not a valid boolean expression within function \"" ^ func.fname ^ "\"") )
          end
    | While(condition, stmts) -> let cond_type = get_expr_type condition func in
      begin
        match cond_type with
            BooleanType ->
              if check_stmt stmts
                then true
              else raise( Failure("Invalid statments in While loop within function \"" ^ func.fname  ^ "\"") )
          | _ -> raise( Failure("Condition of While loop is not a valid boolean expression within function \"" ^ func.fname ^ "\"") )
      end
    | Print(expr) -> let expr_type = get_expr_type expr func in
```

```
      begin
        match expr_type with
            StringType -> true
          | IntType -> true
          | _ -> raise( Failure("Print in function \"" ^ func.fname ^ "\"
            does not match string type") )
      end
    | Draw(_, e1, e2, e3, e4, e5, e6, e7, e8) -> true
  in
    let _ = List.map(fun s -> check_stmt s) func.body in
      true

let valid_func env f =
  let duplicate_functions = function_exist f env in
    (* let duplicate_parameters = count_function_params f in *)
      let v_body = valid_body f env in
        let _ = env.functions <- f :: env.functions (* Adding function to
            environment *) in
        (not duplicate_functions) && (* (not duplicate_parameters) && *)
            v_body

let check_program flist =
  let (environment : env) = { functions = [] (* ; variables = [] *) } in
    let _validate = List.map ( fun f -> valid_func environment f) flist in
        true
```

Listing A.5: Compiler, Code Generation (`compile.ml`)

```
open Ast
open Str
open Printf
open Parser
open Helper
module StringMap = Map.Make(String);;

let string_of_type = function
      IntType -> "int"
    | BooleanType -> "Boolean"
    | StringType -> "String"
    | DoubleType -> "double"
    | _ -> ""
let string_of_var = function
    Molecule(s) -> s

let string_of_element = function
  Element(e) -> e

let string_of_molecule = function
  Molecule(m) -> m

```

```
let string_of_op = function
    Add -> "+"
  | Sub -> "-"
  | Mul -> "*"
  | Div -> "/"
  | Mod -> "%"

let string_of_rop = function
  | Gt -> ">"
  | Geq -> ">="
  | Lt -> "<"
  | Leq -> "<="
  | Eq -> "=="
  | Neq -> "!="

let string_of_re = function
  And -> "&&"
  | Or -> "||"

let string_of_boolean = function
  True -> string_of_bool true
  | False -> string_of_bool false

let string_of_element = function
   Element(e)-> e

let string_of_molecule = function
   Molecule(m)-> m

let string_of_mdecl_balance mdecl = mdecl.mname

let rec string_of_expr = function
  Int(i) -> string_of_int i
  | Double(d) -> string_of_float d
  | Boolean(e1, rop, e2) -> string_of_expr e1 ^ string_of_rop rop ^
    string_of_expr e2
  | String (s) -> s
  | Asn(id, left) -> id ^ " = " ^ (string_of_expr left)
  | Call(s,l) -> s ^ "(" ^ String.concat "" (List.map string_of_expr l) ^ ")"
  | Access(o,m) -> (string_of_expr o) ^ "." ^ m ^"();"
  | Binop (e1, op, e2) ->
  (string_of_expr e1) ^ " " ^ (string_of_op op)
    ^ " " ^ (string_of_expr e2)
  | Brela (e1, op, e2) ->
  (string_of_expr e1) ^ " " ^ (string_of_re op)
    ^ " " ^ (string_of_expr e2)
    | Noexpr -> ""
    | Null -> "NULL"
    | Concat(s1, s2) -> string_of_expr s1 ^ "+" ^ string_of_expr s2
    | List(elist) -> "[" ^  String.concat ", " (List.map string_of_expr elist
        ) ^ "]"
    | Print(s) -> "System.out.println(" ^ string_of_expr s ^ ");"
    | Equation(name, rlist, plist) -> "equation " ^ name ^ "{" ^ String.concat "," (List.map string_of_element rlist) ^ "--" ^ String.concat "," (List.map string_of_element plist) ^ "}"
    | Mass(num) -> num ^ ".mass()"
    | Charge(num) -> num ^ ".charge()"
    | Electrons(num) -> num ^ ".electrons()"
     | Bracket(e) -> "(" ^ string_of_expr e ^ ")"
      | Balance(llist, rlist) -> "Balance(\"" ^ String.concat " , " (List.map string_of_molecule llist) ^ " == " ^ String.concat " , " (List.map string_of_molecule rlist) ^ "\")"


let string_of_edecl edecl = "Element " ^ edecl.name ^ "= new Element(" ^ (string_of_int edecl.mass) ^ "," ^ (string_of_int edecl.electrons) ^ "," ^ (string_of_int edecl.charge) ^ ");\n"
let string_of_mdecl mdecl =  "ArrayList<Element> " ^ mdecl.mname ^ "1 = new ArrayList<Element>(Arrays.asList(" ^ String.concat "," (List.map string_of_element mdecl.elements) ^ "));\n" ^
"Molecule " ^ mdecl.mname ^ "= new Molecule("^ mdecl.mname ^ "1);\n"

let string_of_pdecl pdecl = string_of_type pdecl.paramtype ^ " " ^ pdecl.paramname
let string_of_pdecl_list pdecl_list = String.concat "" (List.map string_of_pdecl pdecl_list)
let string_of_vdecl vdecl = string_of_type vdecl.vtype ^ " " ^ vdecl.vname ^ ";\n"

let rec string_of_stmt = function
      Block(stmts) ->
        "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
    | Expr(expr) -> string_of_expr expr ^ ";\n"
    | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n"
    | If(e, s1, s2) -> "if (" ^ string_of_expr e ^ ")\n" ^ (string_of_stmt s1) ^ "\n" ^ "else\n" ^ (string_of_stmt s2) ^ "\n"
    | For(e1, e2, e3, s) ->
      "for (" ^ string_of_expr e1  ^ " ; " ^ string_of_expr e2 ^ " ; " ^
      string_of_expr e3  ^ ") " ^ string_of_stmt s ^ "\n"
    | While(e, s) -> "while (" ^ string_of_expr e ^ ") {" ^ (string_of_stmt s) ^ "}\n"
    | Print(s) -> "System.out.println(" ^ string_of_expr s ^ ");\n"
    | Draw(s, e1, e2, e3, e4, e5, e6, e7, e8) -> "randx = (int) (Math.random()*400); randy = (int) (Math.random()*400); scene.add(new AtomShape(randx, randy," ^ s ^ "," ^
      (string_of_int e1) ^ "," ^
      (string_of_int e2) ^ "," ^
      (string_of_int e3) ^ "," ^
      (string_of_int e4) ^ "," ^
      (string_of_int e5) ^ "," ^
      (string_of_int e6) ^ "," ^
      (string_of_int e7) ^ "," ^
      (string_of_int e8) ^ "));"


let string_of_vdecl vdecl=
    string_of_type vdecl.vtype ^ " " ^ vdecl.vname ^ ";"

let string_of_fdecl fdecl =
    if fdecl.fname = "main" then "public static void main(String args[])\n{\n
      " ^
  String.concat "" (List.map string_of_vdecl fdecl.locals) ^
  String.concat "" (List.map string_of_edecl fdecl.elements) ^
  String.concat "" (List.map string_of_mdecl fdecl.molecules) ^
  String.concat "" (List.map string_of_stmt fdecl.body) ^
  "}\n"
else
  "public static void " ^ fdecl.fname ^ "(" ^ String.concat ", " (List.map
      string_of_pdecl fdecl.formals) ^ ")\n{\n" ^
  String.concat "" (List.map string_of_vdecl fdecl.locals) ^
  String.concat "" (List.map string_of_edecl fdecl.elements) ^
  String.concat "" (List.map string_of_mdecl fdecl.molecules) ^
  String.concat "" (List.map string_of_stmt fdecl.body) ^
  "}\n"

let string_of_fdecl_list fdecl_list =
    String.concat "" (List.map string_of_fdecl fdecl_list)

let string_of_program (vars, funcs) =
  String.concat "" (List.map string_of_vdecl (List.rev vars) ) ^ "\n" ^
  String.concat "\n" (List.map string_of_fdecl (List.rev funcs) ) ^ "\n"


let rec charge_sum molecule = match molecule with
  | [] -> 0
  | hd :: tl -> hd.charge + charge_sum tl;;


let contains s1 s2 =
    let re = Str.regexp_string s2
    in
        try ignore (Str.search_forward re s1 0); true
        with Not_found -> false

let program program prog_name =
    let graphic_boolean a b =
        if (contains (string_of_fdecl_list program) "graphics") then a else b
            in
     let prog_string = Helper.balance_head ^ prog_name ^ Helper.balance_mid ^
         (graphic_boolean "public static final SceneComponent scene = new
        SceneComponent();" "")  ^ Helper.balance_mid1 ^ prog_name ^ Helper.
        balance_mid15 ^ Helper.balance_mid2 ^ (string_of_fdecl_list program)
         ^ Helper.balance_end in
       let out_chan = open_out (Printf.sprintf "%s.java" prog_name) in
          ignore(Printf.fprintf out_chan "%s" prog_string);
      close_out out_chan;
        ignore(Sys.command(Printf.sprintf "javac %s.java" prog_name));
        ignore(Sys.command(Printf.sprintf "java %s" prog_name));

if (contains (string_of_fdecl_list program) "graphics") then
    let graphics_string = Helper.balance_head ^ "ChemGRAPH extends JFrame" ^
        Helper.balance_mid ^"public static final SceneComponent scene = new
        SceneComponent();" ^ Helper.balance_mid1 ^ "ChemGRAPH" ^ Helper.
        balance_mid15 ^ "setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(500, 500); add(scene, BorderLayout.CENTER);" ^ Helper.
        balance_mid2 ^ (string_of_fdecl_list program) ^ Helper.balance_end in
        let out_chan = open_out ("ChemGRAPH.java") in
            ignore(Printf.fprintf out_chan "%s" graphics_string); close_out
                out_chan;
         (ignore(Sys.command ("javac ChemGRAPH.java SceneEditor.java"));
if (contains (string_of_fdecl_list program) "graphics") then ignore(Sys.
    command("java SceneEditor")));
```

Listing A.6: Helper (`helper.ml`)

```
let balance_head = "import com.graphics.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import javax.swing.*;

public class "

let balance_mid = "{"

let balance_mid1 = "
    public static boolean debug = false;
    public static int randx;
    public static int randy;

    public "

let balance_mid15 = "(){"

let balance_mid2 = "} public static String Balance(String s)
    {
        String output = \"\";
        String[] r = s.split(\"(, )|(==)|(' ')\");
        String[] r1 = s.split(\"\\\\s*(,|\\\\s)\\\\s*\");
        String[] r2 = s.split(\"(, )|(' ')\");
        String[] individual = s.split(\"(, )|(== )|(?=\\\\p{Upper})|(' ')\");

        ArrayList<String> elements = new ArrayList<String>();

        int counter = 0;
        for(int i=0; i<r2.length; i++){
            if(r2[i].contains(\"=\"))
                counter = i;
        }
        counter++;

        for (int i = 0; i < individual.length; i++) {
            String x = \"\";
            for (int j = 0; j < individual[i].length(); j++) {
                if (Character.isLetter(individual[i].charAt(j)))
                    x = x + individual[i].charAt(j);
            }
            if (!elements.contains(x) && (x != \"\"))
                elements.add(x);
        }

        double[][] matrix = new double [elements.size()][r.length];

        for (int i = 0; i < elements.size(); i++) {
            String temp = elements.get(i);
            for (int j = 0; j < r.length; j++) {
                if (r[j].contains(temp)) {
                    int k = r[j].indexOf(temp) + temp.length();
                    if (k >= r[j].length()) {
                        k = 0;
                    }
                    if (Character.isDigit(r[j].charAt(k))) {
                        int dig = Integer.parseInt(r[j].substring(k, k + 1));
                        matrix[i][j] = dig;
                    } else {
                        matrix[i][j] = 1;
                    }
                } else {
                    matrix[i][j] = 0;
                }
            }
        }



        double[][] A = new double[matrix.length][matrix[0].length - 1];
        double[][] B = new double[matrix.length][1];

        for (int i = 0; i < matrix.length; i++) {
            for (int j = 0; j < matrix[i].length - 1; j++) {
                A[i][j] = matrix[i][j];
            }
        }

        int n = A[0].length<A.length? A.length : A[0].length;
        int difference = Math.abs(A.length-A[0].length);
        double[][] A1 = new double[n][n];

        for (int i = 0; i < B.length; i++) {
            B[i][0] = matrix[i][matrix[i].length - 1];
        }


        for(int i = 0; i < A.length; i++)
        {
            for(int j = 0; j < A[0].length; j++)
            {
                A1[i][j] = A[i][j];
            }
        }

        if(A[0].length<A.length){
            for(int i=0; i<n; i++){
                for(int j = n-difference; j< n; j++)
                {
                    A1[i][j] = 1;
                }
            }
        }
        else if (A[0].length> A.length)
        {
            for(int i=0; i<n; i++){
                for(int j = n-difference; j< n; j++)
                {
                    A1[j][i] = 1;
                }
            }
        }

        for(int i=0; i<n; i++)
        {
            for(int j=counter; j<n; j++){
                matrix[i][j] = matrix[i][j] * -1;
            }
        }

        double det = determinant(A1, n);
        double inverse[][] = invert(A1);
        double[][] prod = product(inverse, B, det);

        double factor = 0;
        boolean simplified = true;
        for(int i = 0; i < prod.length; i++)
        {
            for(int j = i; j < prod.length; j++)
            {
                if(mod(prod[i][0],prod[j][0]))
                {
                    simplified = false;
                    break;
                }
            }
        }

        if (simplified == false)
        {
            factor = findSmallest(prod);
            simplify(prod, factor);

        }
        boolean subtract = false;

        for(int j = 0; j < r1.length; j++)
        {
            if(j == r1.length-1)
            {
                int sum = 0;
                int count = 0;
                for(int m = 0; m < B[0].length; m++)
                {
                    if(B[m][0] == 0)
                    {
                        count++;
                    }
                }
                for(int k = 0; k < n; k++)
                {
                    sum += Math.round(matrix[count][k]*Math.abs(prod[k][0]));

                }

                if(B[count][0] == 0)
                {
                    output += 1 + \" \" + r2[j-2];
                }
                else
                {
                    output += Math.abs(sum/(int)B[count][0]) + \" \" + r2[j
                        -2];
                }
            }
            else if(r1[j].equals(\"==\"))
            {
                output += \"--> \";
                subtract = true;
            }
            else if (subtract == true)
            {
                int coeff = (int)Math.round(Math.abs(prod[j-1][0]));
                output += coeff + \" \" + r1[j] + \" \";
            }
            else
            {
                int coeff = (int)Math.round(Math.abs(prod[j][0]));
                output += coeff + \" \" + r1[j] + \" \";
            }
        }
        return output;
    }

    public static boolean mod(double a, double b)
    {

      int c = (int)(a)/(int)(b);
      if (c*b == a)
        return true;
      else
        return false;
    }

    public static void printMatrix(double[][] matrix)
    {
        for (int i = 0; i < matrix.length; i ++)
        {
            for(int j = 0; j< matrix[0].length; j++)
            {
                System.out.print(matrix[i][j] + \" \");
            }
            System.out.print(\"\\n\");
        }
    }

    public static double findSmallest(double a[][])
    {
        double smallest = a[0][0];
        for(int i = 0; i < a.length; i++)
        {
            if(Math.abs(a[i][0]) < Math.abs(smallest))
                smallest = a[i][0];
        }
        return smallest;
    }

    public static double[][] simplify(double a[][], double smallest)
    {
        int largest = 0;
        boolean all = true;
        for(int i = 1; i <=  Math.abs(smallest); i++)
        {
            all = true;
            for(int j = 0; j < a.length; j++)
            {
                if(!mod(a[j][0],i) )
                {
                    all = false;
                }
            }
            if (Math.abs(i)>Math.abs(largest) && all == true)
                largest = i;
        }
        if (debug == true)
            System.out.println(largest);
        if(largest!=0)
        {
            for(int k = 0; k < a.length; k++)
            {
                a[k][0] = a[k][0]/largest;
            }
        }
        return a;
    }

    public static double[][] product(double a[][], double b[][], double det)
    {
        int rowsInA = a.length;
       int columnsInA = a[0].length; // same as rows in B
       int columnsInB = b[0].length;
       double[][] c = new double[rowsInA][columnsInB];
       for (int i = 0; i < rowsInA; i++) {
         for (int j = 0; j < columnsInB; j++) {
             for (int k = 0; k < columnsInA; k++) {
                 c[i][j] = c[i][j] + a[i][k] * b[k][j];
             }
         }
     }

     for(int i = 0; i < rowsInA; i++)
     {
        c[i][0] = c[i][0]*det;
    }
    return c;
}
public static double determinant(double A[][],int N)
{
    double det=0;
    if(N == 1)
    {
        det = A[0][0];
    }
    else if (N == 2)
    {
        det = A[0][0]*A[1][1] - A[1][0]*A[0][1];
    }
    else
    {
        det=0;
        for(int j1=0;j1<N;j1++)
        {
            double[][] m = new double[N-1][];
            for(int k=0;k<(N-1);k++)
            {
               m[k] = new double[N-1];
            }
            for(int i=1;i<N;i++)
            {
                int j2=0;
                for(int j=0;j<N;j++)
                {
                    if(j == j1)
                        continue;
                   m[i-1][j2] = A[i][j];
                    j2++;
                }
            }
            det += Math.pow(-1.0,1.0+j1+1.0)* A[0][j1] * determinant(m,N-1);
        }
    }
    return det;
}
public static double[][] invert(double a[][])
{
    int n = a.length;
    double x[][] = new double[n][n];
    double b[][] = new double[n][n];
    int index[] = new int[n];
    for (int i=0; i<n; ++i)
        b[i][i] = 1;

    gaussian(a, index);

    for (int i=0; i<n-1; ++i)
        for (int j=i+1; j<n; ++j)
            for (int k=0; k<n; ++k)
                b[index[j]][k]
            -= a[index[j]][i]*b[index[i]][k];

            for (int i=0; i<n; ++i)
            {
                x[n-1][i] = b[index[n-1]][i]/a[index[n-1]][n-1];
                for (int j=n-2; j>=0; --j)
                {
                    x[j][i] = b[index[j]][i];
                    for (int k=j+1; k<n; ++k)
                    {
                        x[j][i] -= a[index[j]][k]*x[k][i];
                    }
                    x[j][i] /= a[index[j]][j];
                }
            }
            return x;
        }

// Method to carry out the partial-pivoting Gaussian
// elimination.  Here index[] stores pivoting order.

        public static void gaussian(double a[][], int index[])
        {
            int n = index.length;
            double c[] = new double[n];

 // Initialize the index
            for (int i=0; i<n; ++i)
                index[i] = i;

 // Find the rescaling factors, one from each row
            for (int i=0; i<n; ++i)
            {
                double c1 = 0;
                for (int j=0; j<n; ++j)
                {
                    double c0 = Math.abs(a[i][j]);
                    if (c0 > c1) c1 = c0;
                }
                c[i] = c1;
            }

 // Search the pivoting element from each column
            int k = 0;
            for (int j=0; j<n-1; ++j)
            {
                double pi1 = 0;
                for (int i=j; i<n; ++i)
                {
                    double pi0 = Math.abs(a[index[i]][j]);
                    pi0 /= c[index[i]];
                    if (pi0 > pi1)
                    {
                        pi1 = pi0;
                        k = i;
                    }
                }

   // Interchange rows according to the pivoting order
                int itmp = index[j];
                index[j] = index[k];
                index[k] = itmp;
                for (int i=j+1; i<n; ++i)
                {
                    double pj = a[index[i]][j]/a[index[j]][j];

 // Record pivoting ratios below the diagonal
                    a[index[i]][j] = pj;

 // Modify other elements accordingly
                    for (int l=j+1; l<n; ++l)
                        a[index[i]][l] -= pj*a[index[j]][l];
                }
            }
        }"
let balance_end = "}"
```

Listing A.7: Top-level Executable (`chemlab.ml`)

```
exception NoInputFile
exception InvalidProgram

let usage = Printf.sprintf "Usage: chemlab FILE_NAME"
(* Get the name of the program from the file name. *)
let get_prog_name source_file_path =
  let split_path = (Str.split (Str.regexp_string "/") source_file_path) in
  let file_name = List.nth split_path ((List.length split_path) - 1) in
  let split_name = (Str.split (Str.regexp_string ".") file_name) in
    List.nth split_name ((List.length split_name) - 2)

(* Entry Point: starts here *)
let _ =
  try
    let prog_name =
      if Array.length Sys.argv > 1 then
```

```
        get_prog_name Sys.argv.(1)
      else raise NoInputFile in

    let input_channel = open_in Sys.argv.(1) in

    let lexbuf = Lexing.from_channel input_channel in
      let prog = Parser.program Scanner.token lexbuf in
        if Semantic.check_program prog
          then Compile.program prog prog_name
          else raise InvalidProgram
  with
    | NoInputFile -> ignore(Printf.printf "Please provide a name for a
      ChemLAB file.\n"); exit 1
    | InvalidProgram -> ignore(Printf.printf "Invalid program. Semantic
      errors exist.\n"); exit 1
```

Listing A.8: Test Script (`run.sh`)

```
#!/bin/bash

TESTFILES="files_test/*.chem"
DEMOFILES="files_demo/*.chem"
ran=0
success=0

Compare() {
  diff -bq "$1" "$2" && {
    (( success++ ))
    echo "PASS"
  } || {
    cat "$1"
    echo "FAILED: does not match expected output"
  }
}

for f in $TESTFILES
do
  (( ran++ ))
  name=${f%.chem}      # remove .chem from the end
  name=${name#files_test/}  # remove ./files_test/ from the beginning
  exp=${f%$name.chem}"exp/$name.out"    # insert exp/ into file path
  echo "=========================="
  echo "Testing: $name"
  ./chemlab "$f" > "files_test/$name.out" 2>&1 && {
  # echo "Comparing with $exp"
    # if [[ -e $exp ]]; then
      Compare "files_test/$name.out" "$exp"
    # else
      # echo "FAILED: no expected file found"
    # fi
  } || {
    cat "files_test/$name.out"
    echo "FAILED: did not compile"
  }
done

for f in $DEMOFILES
do
  (( ran++ ))
  name=${f%.chem}      # remove .chem from the end
  name=${name#files_demo/}  # remove ./files_demo/ from the beginning
  exp=${f%$name.chem}"exp/$name.out"    # insert exp/ into file path
  echo "=================="
  echo "Testing: $name"
  ./chemlab "$f" > "files_demo/$name.out" 2>&1 && {
  # echo "Comparing with $exp"
    # if [[ -e $exp ]]; then
      Compare "files_demo/$name.out" "$exp"
    # else
      # echo "FAILED: no expected file found"
    # fi
  } || {
    cat "files_demo/$name.out"
    echo "FAILED: did not compile"
  }
done

echo "=================="
echo "SUMMARY"
echo "Number of tests run: $ran"
echo "Number Passed: $success"
```

Listing A.9: Atom Shape (`AtomShape.java`)

```
package com.graphics;
import java.awt.*;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.*;
import javax.swing.*;
import java.applet.*;
import java.awt.Container;
import java.awt.Font;
import java.awt.RenderingHints;
import java.awt.Shape;
import java.awt.font.FontRenderContext;
import java.awt.font.GlyphVector;
import java.awt.geom.AffineTransform;
import java.awt.geom.*;
import java.awt.image.BufferedImage;
```

```

/*
*{@link CarShape} Creates and handles a CarShape
*@author Cay Horstmann, 2006
*@author Alice Chang, avc2120
*/

public class AtomShape extends CompoundShape
{

  private int centerx;
  private int centery;
  private final int UNIT = 50;
  private int radius;
  private int diameter;
  private int offset = 20;


  public AtomShape(int x, int y, String name, int e_1, int e_2, int e_3, int
      e_4, int e_5, int e_6, int e_7, int e_8)
  {
    super();
    centerx = x;
    centery = y;

    diameter = 1 * UNIT;
    radius = UNIT/2;

    Ellipse2D.Double head = new Ellipse2D.Double(centerx,centery, UNIT, UNIT)
        ;
    //top
    if(e_1 ==1)  {Ellipse2D.Double e5 = new Ellipse2D.Double(centerx + 9 ,
        centery-15, UNIT/4, UNIT/4); add(e5); }
    if(e_2 ==1)  {Ellipse2D.Double e6 = new Ellipse2D.Double(centerx + 29 ,
        centery-15, UNIT/4, UNIT/4); add(e6);}
    //right
    if(e_3 ==1) {Ellipse2D.Double e1 = new Ellipse2D.Double(centerx + 53 ,
        centery+10, UNIT/4, UNIT/4); add(e1);}
    if(e_4 ==1)  {Ellipse2D.Double e2 = new Ellipse2D.Double(centerx + 53 ,
        centery+30, UNIT/4, UNIT/4); add(e2);}
    //bottom
    if(e_5 ==1)  {Ellipse2D.Double e7 = new Ellipse2D.Double(centerx + 9 ,
        centery+53, UNIT/4, UNIT/4); add(e7);}
    if(e_6 ==1)  {Ellipse2D.Double e8 = new Ellipse2D.Double(centerx + 29 ,
        centery+53, UNIT/4, UNIT/4); add(e8);}
    //left
    if(e_7 >=1)  {Ellipse2D.Double e3 = new Ellipse2D.Double(centerx - 15 ,
        centery+10, UNIT/4, UNIT/4); add(e3);}
    if(e_8 >=1)  {Ellipse2D.Double e4 = new Ellipse2D.Double(centerx - 15 ,
        centery+30, UNIT/4, UNIT/4); add(e4);}
```

```java

    Font f = new Font("SansSerif", Font.BOLD, 14);

    BufferedImage img = new BufferedImage(100, 100, BufferedImage.
        TYPE_INT_ARGB);

      Graphics2D g2 = img.createGraphics();

      GlyphVector vect = f.createGlyphVector(g2.getFontRenderContext(), name)
          ;
        Shape shape = vect.getOutline(centerx+20, centery+30);
    add(head);
    add(shape);
  }


}
```

Listing A.10: Compound Shape (`CompoundShape.java`)

```java
  /**
  *{@link CompoundShape}Creates and Handles a CompoundShape Shape
  *@author Cay Horstmann, 2006
  *@author Alice Chang, avc2120
  */
  package com.graphics;
  import java.awt.*;
  import java.awt.geom.*;

  public abstract class CompoundShape extends SelectableShape
  {
    /**
     * Creates a new CompoundShape
     */
     public CompoundShape()
     {
        path = new GeneralPath();
     }

     /**
      * Adds shape s into GeneralPath
      * @param s
      */
     protected void add(Shape s)
     {
        path.append(s, false);
     }

     /**
      * Checks if path contains aPoint
      */
     public boolean contains(Point2D aPoint)
     {
        return path.contains(aPoint);
     }

     /**
      * CHecks if path intersects rectangle
      * @param rec
      * @return
      */
     public boolean intersects(Rectangle2D rec)
     {
       return path.intersects(rec);
     }

     /**
      * translates car by dx and dy
      * @param dx
      * @param dy
      */
     public void translate(double dx, double dy)
     {
        AffineTransform t = AffineTransform.getTranslateInstance(dx, dy);
        path.transform(t);
     }

     /**
      * draws car
      */
     public void draw(Graphics2D g2)
     {
        g2.draw(path);
     }

     private GeneralPath path;
  }
```

Listing A.11: Element (`Element.java`)

```
package com.graphics;
import java.util.Scanner;
import java.util.*;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
import java.util.ArrayList;

```

```java
public class Element
{
  private int charge;
  private int mass;
  private int electrons;
  public Element(int mass, int charge, int electrons)
  {
    this.mass = mass;
    this.charge = charge;
    this.electrons = electrons;
  }

  public int mass()
  {
    return mass;
  }

  public int charge()
  {
    return charge;
  }

  public int electrons()
  {
    return electrons;
  }


}
```

Listing A.12: Molecule (`Molecule.java`)

```java
package com.graphics;
import java.util.ArrayList;

public class Molecule
{
  ArrayList<Element> elements = new ArrayList<Element>();
  public Molecule(ArrayList<Element> element_list)
  {
    elements = element_list;
  }
  public int mass()
  {
    int sum = 0;
    for(int i = 0; i < elements.size(); i++)
    {
      sum += elements.get(i).mass();
    }
    return sum;
  }

  public int charge()
  {
    int sum = 0;
    for(int i = 0; i < elements.size(); i++)
    {
      sum += elements.get(i).charge();
    }
    return sum;
  }

  public int electrons()
  {
    int sum = 0;
    for(int i = 0; i < elements.size(); i++)
    {
      sum += elements.get(i).electrons();
    }
    return sum;
  }
}
```

Listing A.13: Scene Component (`SceneComponent.java`)

```
/**
*{@link SceneComponent}Creates and Handles a SceneComponent
*@author Cay Horstmann, 2006
*@author Alice Chang, avc2120
*/
package com.graphics;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.*;


public class SceneComponent extends JComponent
{
  final Class<AtomShape> AtomShape = AtomShape.class;

  /**
   * Creates a SceneComponent
   */
  private ArrayList<SceneShape> houseCar = new ArrayList<SceneShape>();
  private static final long serialVersionUID = 1L;
  public SceneComponent()
  {
    final Class<AtomShape> AtomShape = AtomShape.class;
    shapes = new ArrayList<SceneShape>();

    addMouseListener(new MouseAdapter()
    {
      public void mousePressed(MouseEvent event)
      {
        mousePoint = event.getPoint();
        for (SceneShape s : shapes)
        {
          if (s.contains(mousePoint))
          {
            s.setSelected(!s.isSelected());
          }
        }
        repaint();
      }
    });

    addMouseMotionListener(new MouseMotionAdapter()
    {
      public void mouseDragged(MouseEvent event)
      {
        Point lastMousePoint = mousePoint;
        mousePoint = event.getPoint();
        for (SceneShape s :shapes)
        {
          if (s.isSelected())
          {
            double dx = mousePoint.getX() - lastMousePoint.getX();
            double dy = mousePoint.getY() - lastMousePoint.getY();
            s.translate((int) dx, (int) dy);
          }
        }
        repaint();
      }
    });
  }


  public void add(SceneShape s)
  {
    shapes.add(s);
    repaint();
  }

  /**
   * @return ArrayList of shapes
   */
  public ArrayList<SceneShape> getShapes()
  {
    return shapes;
  }

  /**
   * @return ArrayList of Selected Shapes
   */
  public ArrayList<SceneShape> getSelected()
  {
    ArrayList<SceneShape> selected = new ArrayList<SceneShape>();
    for (int i = shapes.size() - 1; i >= 0; i--)
    {
      SceneShape s = shapes.get(i);
      if(s.isSelected())
      {
        selected.add(s);
      }
    }
    return selected;
  }

  /**
   * Paints component
   */
  public void paintComponent(Graphics g)
  {
    Graphics2D g2 = (Graphics2D) g;
    for (SceneShape s : shapes)
    {
      s.draw(g2);
      if (s.isSelected())
        s.drawSelection(g2);
    }
  }
  private ArrayList<SceneShape> shapes;
  private Point mousePoint;
}
```

Listing A.14: Scene Shape (`SceneShape.java`)

```
/**
*{@link SceneShape}Creates and Handles a Shape part of a scene
*@author Cay Horstmann, 2006
*@author Alice Chang, avc2120
*/
package com.graphics;
import java.awt.*;
import java.awt.geom.*;

public interface SceneShape
{

  void draw(Graphics2D g2);

  void drawSelection(Graphics2D g2);

  void setSelected(boolean b);

  boolean isSelected();

  void translate(double dx, double dy);
  boolean contains(Point2D p);

}
```

Listing A.15: Selectable Shape (`SelectableShape.java`)

```
/**
*{@link SelectableShape} A shape that manages its selection state.
*@author Cay Horstmann, 2006
*@author Alice Chang, avc2120
*/
package com.graphics;
import java.awt.*;

public abstract class SelectableShape implements SceneShape
{
  /**
   * Sets selected to true or false;
   */
  public void setSelected(boolean b)
  {
    selected = b;
  }

  /**
   * Checks if sceneShape is selected
   * @return true if selected false if not
   */
  public boolean isSelected()
  {
    return selected;
  }

  /**
   * draw selection
   */
  public void drawSelection(Graphics2D g2)
  {
    translate(1, 1);
    draw(g2);
    translate(1, 1);
    draw(g2);
    translate(-2, -2);
  }

  private boolean selected;
}
```

Listing A.16: Scene Editor (`SceneEditor.java`)

```
/*
*@author Alice Chang, avc2120
*/
import com.graphics.*;
public class SceneEditor
{
   public static void main(String[] args)
   {
      ChemGRAPH sceneFrame = new ChemGRAPH();
    ChemGRAPH.graphics();
         sceneFrame.setVisible(true);
   }
}
```

Appendix B

Example Test Code

Listing B.1: Hello World test

```
/* Test 1: Hello World (comments, print) */

function main() {
  print "Hello, world!";
}
```

Listing B.2: Int and String Variable Assignment

```
/* Test 2: int and string variable assignment */

function main() {
  int a;
  int b;
  string s;

  a = 2;
  b = 3;
  s = "ChemLAB";

  print a;
  print b;
  print s;
}
```

Listing B.3: Arithmetic test

```
/* Test 3: Arithmetic Expressions */

function main()
{
  print 0;
  print 1;

  /* Plus, minus, multiply, divide, mod */
  print 1+1;
  print 4-1;
  print 2*2;
  print 15/3;
  print 41%7;

  /* Precedence */
  print 90-6*8;

  /* Parenthesis */
  print (1+2*3-4)*28/2;

  /* Negative Numbers */
  // print -3-39;
  // print 14*-3;

  /* Decimals */
  // print 2.1;
  // print 5.0/2.0;
  // print 42.0/99.0;
}
```

Listing B.4: String Concatenation

```
/* Test 4: String concatenation */

function main()
{
  string a;
  string b;
  string c;
  a = "Hello";
  b = "world";
  c = "!";
  print a ^ ", " ^ b ^ c;
}
```

Listing B.5: If Condition

```
/* Test 5: If Conditional, Boolean */

function main() {
  int x;
  int y;
  x = 17;
  y = 42;
  if (x < y) {
    print x ^ " is less than " ^ y;
  } else {
    print "Test Failed";
  }
}
```

Listing B.6: Nested If Condition

```
/* Test 6: Nested If Else */

function main()
{
  int x;
  int y;
  x = 17;
  y = 39;

  if (x != y) {
    y = y + 2;

    if (x > y) {
      print "Inner If Failed";
    } else {
      y = y + 1;
    }
  } else {
    print "Outer If Failed";
  }

  print y;
}
```

Listing B.7: While Loop

```
/* Test 7: While Loop */

function main()
{
  int a;
  int b;
  a = 0;
  b = 3;


  while(a < b)
  {
    a = a + 1;
    print a;
  }
}
```

Listing B.8: Draw

```
function main()
{
  int a;
  element C(12,13,14);
  a = C.mass;
  print a;

}
```

Listing B.9: Balance

```
function main ()
{
  string eq;
  element C(12,12,12);
  eq = balance(HNO3, Cu1 --> CuN2O6, H2O, NO);
  print eq;
}
```

Listing B.10: Demo 1

```
function main()
{
  print "Hello World";
  call message("PLT rocks");
}
function message(string m) {
  print m;
}
```

Listing B.11: Demo 2

```
function a()
{
  string a; string b; string c;
  string d; string e; string f; string g;
  a = balance(MgO, Fe1 --> Fe2O3, Mg1);

  b = balance(Mg1, HCl --> MgCl2, H2);

    c = balance(Cl2, CaO2H2 --> CaCl2O2, CaCl2, H2O);

    d = balance(HNO3, Cu1 --> CuN2O6, H2O, NO);

    e = balance(C3H8O, O2 --> CO2, H2O);

    f = balance(KBr, KMnO4, H2SO4 --> Br2, MnSO4, K2SO4, H2O);

    g = balance(HNO3, Cu1 --> CuN2O6, H2O, NO);

    print a;
    print b;
    print c;
    print d;
    print e;
    print f;
    print g;
}
function main()
{
  call a();
}
```

Listing B.12: Demo 3

```
function calculatekc()
{
  int sum;
  string eq;
  element S (8,16,2);
  element Br (16, 32, 2);
  element Mn (25, 55, 2);
  element O (8, 16, 2);
  element K (19, 39, 1);
  element H (1, 2, 1);

  molecule KBr {K, Br};
  molecule KMnO4 {K, Mn, O, O, O, O};
  molecule H2SO4 {H, H, S, O, O, O, O};

  eq = balance(KBr, KMnO4, H2SO4 --> Br2, MnSO4, K2SO4, H2O);
  print eq;

}
function graphics()
{
  draw("K",1,1,1,1,1,1,1,0);
  draw("Br",0,0,0,1,0,0,0,0);
  draw("Mn",1,1,0,0,0,0,0,0);
  draw("O",1,1,1,0,1,1,1,0);
  draw("H",1,0,0,0,0,0,0,0);
  draw("H",0,1,0,0,0,0,0,0);
}
function main()
{
  call calculatekc();
}
```

Listing B.13: Demo 4

```
function reduction()
{
  string bal;
    int reactants;
    int products;
    int reactants_total;
    int products_total;

    element Fe (26,56,2);
    element Cl (17, 35, 1);
    element Zs (30, 65, 0);   // Zs is Zinc solid
  element Zn (30, 65, 2);

  molecule FeCl2 {Fe, Cl, Cl};
  molecule ZnCl2 {Zn, Cl, Cl};

    bal = balance(FeCl2, Zn1 --> ZnCl2, Fe1);

    reactants = Zn.charge;
    products = Zs.charge;

    reactants_total = mol_charge(FeCl2) + mol_charge(Zn);
    products_total = mol_charge(ZnCl2) + mol_charge(Fe);

    print "Charge of Zn Reactant: " + reactants;
    print "Charge of Zn Product: " + products + "\n";

  if(reactants > products)
  {
    print "Fe is oxidized and Zn is reduced";
    print "Fe is the reducing agent and Zn is the oxidizing agent\n";
  }
  else
  {
    print "Zn is oxidized and Fe is reduced";
    print "Zn is the reducing agent and Fe is the oxidizing agent\n";
  }

  print "Checking if balanced...\n";

  if(reactants_total == products_total)
  {
    print "Yes it is balanced! :)";
  }
  else
  {
    print "Not balanced :(";
  }

}

function main()
{
  print "Calculating Redox Formula...\n";
  call reduction();
}
```

Appendix C

Project Log

Listing C.1: Project Log from GitHub

```
commit fa253c7e160d87fa61ede5ec10b475e24c273a19
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 17 23:18:02 2014 -0500

    ALL DONE!

commit 14365cc6d5e833c5d5df2380192e8135994e30c9
Author: detectiveconan2 <detectiveconan2@users.noreply.github.com>
Date:   Wed Dec 17 23:03:36 2014 -0500

    final

commit 1f772975d571c53ed175dff80af25550d6b65be5
Author: detectiveconan2 <detectiveconan2@users.noreply.github.com>
Date:   Wed Dec 17 22:53:46 2014 -0500

    final

commit 96ac3cae571906048f688dcd859c203868aaa080
Merge: 2991a32 fa2b80c
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 17 21:42:15 2014 -0500

    Merge branch 'Martin-3'

    Conflicts:
      parser.mly
      scanner.mll
      semantic.ml

commit fa2b80c72d4d209a415f9833edc3359e7474a64b
```

```
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 17 21:40:15 2014 -0500

    Update stuff

commit 2991a3234d2ba13ff4894c306b993b7fd6fc9e02
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 17 17:30:39 2014 -0500

    Remove output files

commit c46f9e47e4094ae49754173d29d333f63a558bdc
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 17 17:29:52 2014 -0500

    Update Makefile

commit f66a1fad1c16d0d574e9944c2e54bfed0cbe05ec
Merge: 9ddba36 06fb1ff
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 17 17:27:10 2014 -0500

    Merging with Martin

commit 9ddba36deb0f4a4345729f7319fcd99bb333dcb3
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 17 13:17:13 2014 -0500

    Working

commit 06fb1ffc22524c170ea49168ae2f11bf4872c231
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 17 11:00:04 2014 -0500

    demo and tests work

commit 1767e6342cf119125f0760092d39ef7c839dce58
Merge: ecfe834 e5f6154
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 17 10:10:33 2014 -0500

    Merge branch 'Alice-6'

commit e5f61546554d9a113899cd8007db78306dcee9b4
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 17 10:10:22 2014 -0500

    all tests work

commit a3a7e7226da585e859b2a6b5e8439e5e1c5c307a
```

```
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 17 09:54:00 2014 -0500

    demos work

commit ecfe834c427e0c3c4112281a473994f5fe70de85
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 17 05:36:37 2014 -0500

    Modified test files

commit 25868b3a775ded55749d3aaa695f4b0c785feb94
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 17 05:33:52 2014 -0500

    Did lots of stuff. Improved semantic check, parsing, ast, etc.

commit 1b2be93217eb6da33fe545198b317b43b3ce43b0
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 17 03:12:28 2014 -0500

    Update file structure for tests

commit 534f12a28d362f20a09bdd8d54a4dc0c9f5e786b
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 17 02:54:07 2014 -0500

    Resolve conflicts with merge

commit 21695f006dfd2a2184b31327154abcfa4029dc5f
Merge: 058c11a d0eefdf
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 17 02:46:17 2014 -0500

    Merge branch 'master' of https://github.com/martinong/ChemLAB

    Conflicts:
      files_test/CodeTest/test_element.chem
      files_test/CodeTest/test_element_molecule.chem
      files_test/CodeTest/test_function_call.chem
      files_test/CodeTest/test_function_call.out
      files_test/SemanticTest/semantictest1.chem
      files_test/test3.chem
      run.sh
      semantictest1.chem
      test/CodeTest/test_element.chem
      test/CodeTest/test_element_molecule.chem
      test/CodeTest/test_function_call.chem
      test/CodeTest/test_function_call.out
      test/SemanticTest/semantictest1.chem
```

```
        test/exp/test_function_call.out
        test/test_element.chem
        test/test_element_molecule.chem
        test/test_function_call.chem

commit 058c11a00ff54aa34ed24668b608bcfcace217cb
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 17 02:36:55 2014 -0500

    Updated tests structure

commit d0eefdf6a1afa2a63ff9436dd697d6606c1f12ec
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 17 02:33:40 2014 -0500

    commented out CodeTest in run.sh

commit a5d00eb38853180b3f5aaf7d71957f7ed1170e47
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 17 02:18:56 2014 -0500

    changed file names so can rm files easily in Makefile

commit 3c83a176de44198c0bc45680e6fee38c8f9ce40a
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 17 02:09:24 2014 -0500

    demos

commit 0a76a7459b7b419d22982387783eae6fddc2596c
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 17 01:20:58 2014 -0500

    moved demo4

commit 6c69025ace3a43b27e2b95cf3e806cb927e5ca3f
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 17 01:20:15 2014 -0500

    demo

commit a0c19c312b2b4ef4a4deff7792780fa46f8ea865
Merge: ed9a7d9 7ccba08
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 17 01:16:59 2014 -0500

    Merge branch 'master' of https://github.com/martinong/ChemLAB

    Conflicts:
      SceneEditor.java
      compile.ml
      test/test9.chem

commit ed9a7d9aa4dc6926e482f06ab21dc61fc4867b66
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 17 01:15:56 2014 -0500

    demos added

commit 7ccba08a70336eb9b3bd1f378ced0f0606d8e370
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 17 00:59:52 2014 -0500

    Added project log to Final Report

commit c7fabe40268ff2836ce5f7567921cc420a235762
Author: Alice Chang <avc2120@columbia.edu>
Date:   Tue Dec 16 23:05:10 2014 -0500

    deleted rule

commit 2a40d291f771f870f403f17f44c908514ac7aa95
Author: Martin Ong <mo2454@columbia.edu>
Date:   Tue Dec 16 22:58:11 2014 -0500

    Update test cases

commit a19a452c8dd8d4606178e73f5adcb52cde682c31
Author: Martin Ong <mo2454@columbia.edu>
Date:   Tue Dec 16 22:57:57 2014 -0500

    Changed a lot of stuff

    Tests changed

commit 9e6dbe79595b5f5179080e91f09a1116ad3e6297
Merge: e6f2c1d acc4b94
Author: Alice Chang <avc2120@columbia.edu>
Date:   Tue Dec 16 21:53:25 2014 -0500

    Merge branch 'Alice-5'

commit acc4b94ef88607effb602d968b1afa35bc9ef0e5
Author: Alice Chang <avc2120@columbia.edu>
Date:   Tue Dec 16 21:53:09 2014 -0500

    balance parses

commit 49d87a8ac2a48745393046a58e73299a1df64047
Author: Alice Chang <avc2120@columbia.edu>
```

```
Date:   Tue Dec 16 20:41:45 2014 -0500

    helper added

commit e6f2c1d800afa90c5ff2ca2003b351249bbd453f
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Dec 15 21:23:52 2014 -0500

    Outputs error when it doesn't match file for test

commit d4913a22046441b46dca2c74e2897f26cfc0ed9c
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Dec 15 21:08:58 2014 -0500

    Delete test7

commit 1bf6acc7a9444876c63dbb1fc67afda06f040ade
Merge: df2360e 63ff646
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Dec 15 21:08:38 2014 -0500

    Merge branch 'Semantic-2'

    Conflicts:
      ast.mli
      compile.ml
      parser.mly
      scanner.mll
      semantic.ml
      test/test6.chem

commit 63ff64621063684e4581b22aeddb8703d1927a8f
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Dec 15 21:02:05 2014 -0500

    Fix test cases, no parse errors

commit e0a6035f24b233601584b89a2a383fecd59e23d8
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Dec 15 20:58:01 2014 -0500

    Added single line comments

commit 7046b6e22e5fdb85dc36291b1ff8867cf8c224c6
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Dec 15 20:43:02 2014 -0500

    Removed duplicate file

commit e108acbc8e2cd7427f4c774fa42dd97e90894f21
```

```
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Dec 15 20:41:44 2014 -0500

    Updated make clean

commit 729e19f4b38c289bd9be713adc0a063de25e7c40
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Dec 15 20:35:03 2014 -0500

    Updated test script

commit 025c95cbe8191abae9cf8cd45254a4ac06dcf683
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Dec 15 19:02:46 2014 -0500

    Update parser, scanner, ast to include mod, concat

commit b2f53f3d6cde3a5a68aebbfe5f3d585452f14d80
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Dec 15 19:01:44 2014 -0500

    Check tests against expected

    Put expected outputs in folder exp

commit 14d6922b3a0699b4eaf8aabdb77180333c751350
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Dec 15 17:48:57 2014 -0500

    Updated all test files

commit 222606b774b902e111ca0ce5f7829d981d048a90
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Dec 15 17:48:32 2014 -0500

    Updated all test files

commit df2360ecf133f33537d4c7b9cbf202f1eb281846
Author: Martin Ong <martinong@users.noreply.github.com>
Date:   Mon Dec 15 14:43:13 2014 -0500

    Update README.md

commit 6598a0242fa8b231aa1f0a2a3089a039aa426339
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sun Dec 14 23:43:06 2014 -0500

    Formatted code listings

commit 9f9678207a2e467ab461a0361773734ea4481c7c
```

```
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sun Dec 14 23:16:05 2014 -0500

    Added image in

commit 366b2fe0e0d3c183e2719837986a8edcc3bd7d5d
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sun Dec 14 23:10:07 2014 -0500

    Restructured file structure

commit 70b301b99934cbc86ccaf65fc229c86112b8a173
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sun Dec 14 23:07:28 2014 -0500

    Initial Commit for Final Report in Latex

commit ac5847f846e619b2a533f3233d146b1a673d1afb
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Sun Dec 14 23:05:00 2014 -0500

    picture

commit e58efc348f0952ac3e2aef1e2278e609f41667db
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Sun Dec 14 23:03:49 2014 -0500

    final paper

commit f40dbf7d0d44c68ce772ef004c371b335dfb6bf3
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Sun Dec 14 22:18:27 2014 -0500

    final paper

commit 9f69d7e7493e8c693921b39124280b94c7dbce56
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Sun Dec 14 21:33:55 2014 -0500

    final paper parts

commit 3b65b7b4142d1f4243c74a23c8968d991e25fbb4
Author: Alice Chang <avc2120@columbia.edu>
Date:   Sun Dec 14 15:34:36 2014 -0500

    deleted contents of ChemLAB.java

commit 17f9a2020385a6966bcb8367f22c0e1aa75abd2f
Merge: 299e376 1c2d5c9
Author: Alice Chang <avc2120@columbia.edu>
```

```
Date:   Sun Dec 14 13:15:16 2014 -0500

    Merge branch 'Alice--2'

    Conflicts:
      compile.ml
      parser.mly

commit 299e37699aec217bffacfe28b89e89a383c105a1
Merge: 0f76289 f8168c2
Author: Martin Ong <martinong@users.noreply.github.com>
Date:   Sun Dec 14 11:28:08 2014 -0500

    Merge pull request #5 from martinong/Semantic-2

    Major Debug

commit 0f76289e7697f8ff8fbe95d88b3a0bfc0bd95e7b
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sun Dec 14 11:24:45 2014 -0500

    Merge branch 'Semantic-2'

commit f8168c2443120ea304a5c48f163fce9e295251a8
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sun Dec 14 02:02:05 2014 -0500

    Major Debug

    If, For, While loops fix
    Concat works

commit 6d4cc1a94356b409ed1c803f4ad6f1dc2df2f05c
Merge: 141c700 c6c3293
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sun Dec 14 01:06:23 2014 -0500

    Merge branch 'Martin-Semantic'

    Conflicts:
      ChemLAB.class
      chemlab.ml
      compile.ml
      semantic.ml

commit c6c329373c317929593db4c5fa44a7492e83082e
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sun Dec 14 00:56:34 2014 -0500

    Remove Element, Molecule and Equation types
```

```

commit 2f8a8086f101c5a85d3f4abc8331598471cdc264
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sun Dec 14 00:50:28 2014 -0500

    Add string_of_type to return the java string

commit c4863f980a81d9dd7befb04fce2fbab3de909533
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sun Dec 14 00:41:14 2014 -0500

    Change type from a string to "data_type"

commit f70c574f9a82028852aa8e83b7e50e7aa53cc755
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sun Dec 14 00:11:23 2014 -0500

    Semantic check compiles

    Checks for valid body. Still buggy (Maybe problems with get_expr_type)

commit fe0c5a310de9125a27f4e109619209190d1f3403
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sun Dec 14 00:10:18 2014 -0500

    Update spacing of test file

commit 1c2d5c9f611fa7ca532d4f6d99f32acdd724fa66
Author: Alice Chang <avc2120@columbia.edu>
Date:   Sat Dec 13 21:37:42 2014 -0500

    works

commit e4a11cc5944a5a0d2da73b44351c92c168403e71
Author: Alice Chang <avc2120@columbia.edu>
Date:   Sat Dec 13 18:55:23 2014 -0500

    readme edited

commit 7b50d47da002ff48c9ba56245d3cd074fe8858a6
Author: Alice Chang <avc2120@columbia.edu>
Date:   Sat Dec 13 18:51:45 2014 -0500

    cleaned up

commit 430fb2ea4d8aa1f27e5e9774504b636e8c6d71fb
Merge: 5de8527 46b797c
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Sat Dec 13 18:35:51 2014 -0500

```

```
    Merge remote-tracking branch 'origin/Martin-Semantic' into Martin-
        Semantic

    Conflicts:
      semantic.ml

commit 46b797cb8cf1e386957a1db01fc5ac3fcb42dab3
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sat Dec 13 18:34:31 2014 -0500

    Validate Body statements

commit 5de85273bf243307cdeceb78820e42dbf12176dd
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Sat Dec 13 18:33:47 2014 -0500

    updated semantic

commit 141c700104e66341f7c54198c9aa1a7dbeb0a7bf
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Sat Dec 13 18:29:15 2014 -0500

    semantic-check exp

commit a4d54011235fa544a7f57a86a4ad89772986b65b
Author: Alice Chang <avc2120@columbia.edu>
Date:   Sat Dec 13 18:24:53 2014 -0500

    fixed

commit 73e696882a285c9552ad94845d4b347ce941f548
Author: Alice Chang <avc2120@columbia.edu>
Date:   Sat Dec 13 18:19:45 2014 -0500

    only does graphics if needs

commit 96abeb25b77b6918b70ba49808edd2d5c0f34f10
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Sat Dec 13 17:31:48 2014 -0500

    semantic - add if else

commit 1816cada36af54d7f15e47ea29317212317e8b39
Author: Alice Chang <avc2120@columbia.edu>
Date:   Sat Dec 13 16:41:08 2014 -0500

    graphics works

commit a961d8d2b9d24cde506318675ff8e7612e9bf328
Author: Alice Chang <avc2120@columbia.edu>
```

```
Date:   Sat Dec 13 14:18:30 2014 -0500

    Added Graphics

commit 9638709611d3f8a44fb68e6c2f75719db60099db
Author: Alice Chang <avc2120@columbia.edu>
Date:   Fri Dec 12 16:09:45 2014 -0500

    atom name added

commit 8d689b57835fdfbb51147b6153699980ecec9b2c
Author: Alice Chang <avc2120@columbia.edu>
Date:   Fri Dec 12 14:41:02 2014 -0500

    added access

commit 40e7c4c05fc091c84b9905a0e076090b124dfba7
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Thu Dec 11 16:19:49 2014 -0500

    more type checking-semantic

commit c1a55300d13fa05fceb6e30f063e35549d44a096
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Thu Dec 11 15:39:32 2014 -0500

    semantic can check type of formals and vars

commit 3770dfa639145856494eeaa98356dc729ecf0f25
Author: Martin Ong <mo2454@columbia.edu>
Date:   Thu Dec 11 15:36:29 2014 -0500

    Semantic check function parameters

commit d4b62ce73837a83e44475d3b1bb851033649bf65
Author: Alice Chang <avc2120@columbia.edu>
Date:   Thu Dec 11 14:44:44 2014 -0500

    test8

commit 4061602fb31778cc3a894b91fe4a27f334827cf8
Author: Alice Chang <avc2120@columbia.edu>
Date:   Thu Dec 11 14:44:14 2014 -0500

    added while and test

commit 772979df7eb1f8fd4938aa218306bc653f12128e
Merge: b9e106a 4665f8b
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Thu Dec 11 14:13:43 2014 -0500
```

```

    semantic

commit b9e106ad836a537a7c8b8839bb5ec1736f0106e4
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Thu Dec 11 14:12:00 2014 -0500

    semantic - one error

commit 4665f8b74515c207e8f53801222aa35bab4ab13b
Merge: 65d1725 c448186
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 10 22:09:38 2014 -0500

    Merge branch 'Alice-1'

commit c4481864bb83166f93ca80348439f3b5dda0b454
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 10 22:08:34 2014 -0500

    Compiler Done!

commit 25b9f6191d3f6e6c4f97a67c82b450ca63be7d82
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 10 20:09:32 2014 -0500

    Update gitignore to exclude .java files

commit b0a25d3a3c8073b076ef1dde5251a292ec8a84f9
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 10 15:41:01 2014 -0500

    hello world compiles!

commit 65d17258dfe5e71bf9fbf2a411ed4008629e6e6e
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 10 15:13:33 2014 -0500

    Merge branch 'Hello-World'

    Conflicts:
      ast.mli

commit 71b5686c9a48c57d1ad1359bdea07e674055ab2b
Author: Martin Ong <martinong@users.noreply.github.com>
Date:   Wed Dec 10 14:44:35 2014 -0500

    Update .gitignore

    Ignore test files generated by compiler
```

```

commit f36895bb0df639376bae04cbfd62ee3af3d35d29
Merge: 8f1698b af20856
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 10 14:44:12 2014 -0500

    Merge branch 'master' of https://github.com/martinong/ChemLAB

commit 8f1698b02b35965390ea9a0c6f0b54520821008d
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 10 14:44:10 2014 -0500

    deleted test files

commit 290b4962d82b491f2a709d216d760ce9fd3d53eb
Merge: 442fc78 24048cb
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 10 14:43:42 2014 -0500

    Merge branch 'master' of https://github.com/martinong/ChemLAB

    Conflicts:
      ast.mli

commit af208566e2650dad6b689807448d7c6ef548df15
Merge: 24048cb df403a5
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 10 14:36:24 2014 -0500

    Merge branch 'Martin'

commit df403a53220400f9965d9552a5e8ecbb1937d251
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 10 12:39:02 2014 -0500

    Compiles and runs balancing one equation

commit 2d933805ef58312bce84937c4fdc8fa03c9f74d2
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Dec 10 12:24:19 2014 -0500

    Change main class name from "ChemLAB"

commit 24048cb02b5dafa8fae9e46bb61cdf7fad0cebda
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sat Dec 6 02:01:51 2014 -0500

    Minor touch ups

commit 4946d0dfad380cb72aff8e9cf8dc4a62fc9bef5f
```

```
Author: Martin Ong <martinong@users.noreply.github.com>
Date:   Sat Dec 6 02:00:49 2014 -0500

    Delete chemistry.class

commit 06b9a52f11e9be2b03a9f96773490d3168bf8dbe
Author: Martin Ong <martinong@users.noreply.github.com>
Date:   Sat Dec 6 02:00:41 2014 -0500

    Delete ChemLAB.class

commit ead2bf34ba65795ef9c77bf0195ad72981d8c1c3
Merge: e2410d9 a3a1787
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sat Dec 6 01:55:14 2014 -0500

    Merge branch 'Hello-World'

    Conflicts:
      .gitignore
      ChemLAB.class
      chemistry.class
      compile.ml

commit e2410d9af6095fb2fee86053f08bce20694ad141
Merge: 93fcef3 ea0a11e
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sat Dec 6 01:39:47 2014 -0500

    Merge branch 'Gabe'

    Conflicts:
      Parse.java
      ast.mli
      semantic.ml

commit 93fcef39b26203931fec19e903e0154d2b345280
Merge: c0ce58b d8bec35
Author: Martin Ong <martinong@users.noreply.github.com>
Date:   Sat Dec 6 01:29:27 2014 -0500

    Merge pull request #1 from martinong/Martin

    Martin

commit d8bec354554c0bd18d61272caf8edca1e0de5e6e
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sat Dec 6 01:27:32 2014 -0500

    Changed "chemlab.ml" so that it can take arguments
```

```

commit ea0a11efe340c6d1affacaa45560ae1253fffd16
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Sat Dec 6 00:01:55 2014 -0500

    semantic compiles now

commit b9107a8684d79759cd09f4e19b32cebb90ee8c3f
Author: Martin Ong <mo2454@columbia.edu>
Date:   Fri Dec 5 23:06:26 2014 -0500

    Make test.sh fancier

commit a3a1787fd9f556e3c509d3f39bbd7047881d70b3
Author: Alice Chang <avc2120@columbia.edu>
Date:   Fri Dec 5 23:02:35 2014 -0500

    outputs equation

commit 442fc7809ee3bb259903b07d64e313851c026355
Author: Alice Chang <avc2120@columbia.edu>
Date:   Fri Dec 5 22:56:38 2014 -0500

    prints out equation from compiler

commit c0ce58b6fd1241e14fc28d71535addbc635e8f7b
Author: Martin Ong <mo2454@columbia.edu>
Date:   Fri Dec 5 22:33:51 2014 -0500

    Updated so that *.class is cleaned in "make clean"

commit 567834da2f2a67c1dc553f7dfad9290afb1e4837
Author: Martin Ong <mo2454@columbia.edu>
Date:   Fri Dec 5 22:28:40 2014 -0500

    Ignore *.class files

commit acafa50b818fd91608cdd0d55cd6baf0aab0efe4
Merge: 7cb40f0 fe5a59a
Author: Martin Ong <mo2454@columbia.edu>
Date:   Fri Dec 5 22:23:57 2014 -0500

    Merge branch 'Hello-World'

commit 531f8c2d1fb79436366258d0c1f75e1c7509b5f3
Author: Alice Chang <avc2120@columbia.edu>
Date:   Fri Dec 5 19:32:28 2014 -0500

    fixed

```

```
commit fe5a59ac6382179d1860157455ad80be309acdf7
Author: Alice Chang <avc2120@columbia.edu>
Date:   Thu Dec 4 16:59:18 2014 -0500

    edited!

commit 47093cb30085385a1f5c9ca92fd7397468c7fb2c
Author: Alice Chang <avc2120@columbia.edu>
Date:   Thu Dec 4 16:44:37 2014 -0500

    error free!

commit f2befff33231a7638e9cee0945fcf46f557b59d5
Author: Alice Chang <avc2120@columbia.edu>
Date:   Thu Dec 4 14:56:21 2014 -0500

    Fixed chemistry.java

commit cde87eb4db67e168cfe8cf45a7072e1adc93fa1f
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Thu Dec 4 00:29:17 2014 -0500

    function for CHEMLAB

commit 2ac58c076c8e6931694a2e2adff48b9079b2c1cf
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Dec 3 20:07:09 2014 -0500

    makefile changed and parser fixed

commit 1067f2c04e1f2b524ddba20545350861acd3c12f
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Wed Dec 3 00:02:01 2014 -0500

    semantic analyzer, added some checking for func

commit ca835111c8227af4590dfb3480b5fbe16eb92f2a
Author: Alice Chang <avc2120@columbia.edu>
Date:   Tue Dec 2 18:07:26 2014 -0500

    compiler

commit b2010c8ffaad346d291fa0f80b5914ddab94ac4f
Author: Alice Chang <avc2120@columbia.edu>
Date:   Tue Dec 2 15:16:17 2014 -0500

    added files and java program

commit d08499229e2538b818567096475f7ba0f8a67239
Author: Alice Chang <avc2120@columbia.edu>
```

```
Date:   Sun Nov 30 14:42:06 2014 -0500

    added semantic check

commit b1d3aa91e1c93d02203ce6f89adc9b85ff58eadf
Author: Alice Chang <avc2120@columbia.edu>
Date:   Sun Nov 30 02:08:37 2014 -0500

    first draft of parser done and working

commit 7a08f7b6cea8d7769a495c82e92634f5de3ad6a1
Author: Alice Chang <avc2120@columbia.edu>
Date:   Sun Nov 30 02:02:47 2014 -0500

    fixed equation and molecule

commit 5bf35f528137afcedc3354dbccc5b233c334647d
Author: Alice Chang <avc2120@columbia.edu>
Date:   Sun Nov 30 01:57:25 2014 -0500

    fixed elements and molecules

commit b98a5eebd57850d3895717de441afad6f8c60a06
Author: Alice Chang <avc2120@columbia.edu>
Date:   Thu Nov 27 13:50:08 2014 -0500

    edited ast

commit dc74852b700b71ab04374785303d3f435bf958b3
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Nov 26 16:51:14 2014 -0500

    Merged

commit 0cf4ad2cc11cb83ef5b8f9dc30033fb85b9fb6c9
Merge: b8845ed ca1659e
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Nov 26 16:42:40 2014 -0500

    Merge branch 'master' into Hello-World

    Conflicts:
      ast.mli
      chemlab.ml
      parser.mly
      scanner.mll
      test2.chem
      test3.chem

commit b8845ed2c416bb5e628d4b2a83f2865737ba578e
```

```
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Nov 26 16:40:03 2014 -0500

    Changed Makefile and Ast

commit ca1659ec93bfc1522b0bf7aa6bf4264130d860fe
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Nov 26 16:15:08 2014 -0500

    Test cases work

commit fdd1517b2781e78ca4f2987004722ec5b54df4ed
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Nov 26 11:39:05 2014 -0500

    added function functionality

commit fa99248d5d95e6c31576419b9ffbb25a7138976a
Author: Alice Chang <avc2120@columbia.edu>
Date:   Tue Nov 25 16:01:02 2014 -0500

    All test cases work

commit 04732499b6c2b5f61f13563b6613db6670c5559f
Author: Alice Chang <avc2120@columbia.edu>
Date:   Tue Nov 25 15:39:51 2014 -0500

    Added And Or

commit f40f7f6bb740370b5ed98ad719d146958300c38a
Author: Alice Chang <avc2120@columbia.edu>
Date:   Tue Nov 25 15:31:54 2014 -0500

    All test cases working from 1-9 exempt 2

commit 0ca59079eae956fbbd1021e7a2b84f4147f7fd24
Author: Alice Chang <avc2120@columbia.edu>
Date:   Tue Nov 25 15:12:17 2014 -0500

    Conditional and Arithmetic Working

commit cea344f925510da17345eaad652a6e6e5185ce6e
Author: Alice Chang <avc2120@columbia.edu>
Date:   Tue Nov 25 14:02:28 2014 -0500

    Equation Declaration Works

commit e3495ba8aaae28c0462a81842bc663effe4a9e51
Author: Alice Chang <avc2120@columbia.edu>
Date:   Tue Nov 25 13:42:16 2014 -0500
```

```

    molecule declaration works

commit 7cb40f049e0a22498febc81c92ef7d9f9ddca1d7
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Nov 24 22:58:59 2014 -0500

    Arithmetic parsed

    Work on statement lists

commit d00a90a5f9d1298d3e6c493f261bf8a47c7d2cd2
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Nov 24 21:50:03 2014 -0500

    Parser for test2

commit 88ae50e8901e9a7048066afca003e0a0f666576c
Author: Alice Chang <avc2120@columbia.edu>
Date:   Mon Nov 24 21:49:16 2014 -0500

    sat test2 done

commit 234a653fac8ced141e6dd814ed40698e281374f9
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Nov 24 21:30:21 2014 -0500

    Hello World!

    Able to parse hello world test (test1)

commit 7503e6c2f58d566ba96b33868f4c16444a2e2795
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Nov 24 21:22:50 2014 -0500

    This makes now

commit bbfeb62c86c41fbd985f70199c891ecd5d6a78f9
Author: Alice Chang <avc2120@columbia.edu>
Date:   Mon Nov 24 19:12:50 2014 -0500

    Original

commit dc1dae4471accb5ed2074d540c1bd99fa554c8f1
Author: Alice Chang <avc2120@columbia.edu>
Date:   Mon Nov 24 18:34:40 2014 -0500

    deleted fdecl

commit 9d2415ffa9b0cc5d8fc0a257890af73855dcb906
```

```
Author: Alice Chang <avc2120@columbia.edu>
Date:   Mon Nov 24 18:25:57 2014 -0500

    errors fixed

commit 25d667606ff66750854fbd54c314caf55b8a3057
Author: Alice Chang <avc2120@columbia.edu>
Date:   Mon Nov 24 18:22:28 2014 -0500

    edited parser element

commit 16ef686ec8d9e691f54a1f242fc45bcaa6e54219
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Nov 24 18:18:49 2014 -0500

    Fixed test cases to include data type declaration

commit 9cb52f71c1fd94f11618d8444a93b6b46f134d45
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Nov 19 21:53:47 2014 -0500

    Debug

commit 5e0ddc6e268304f8ad7676148b723007dd887ee4
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Nov 19 21:13:16 2014 -0500

    Debug

commit 2bdd02c48a3464098e113e66bb8132748a65a75e
Merge: e0eefa4 04d0f25
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Wed Nov 19 21:12:06 2014 -0500

    Merge remote-tracking branch 'origin/master'

    Conflicts:
      ast.mli

commit 04d0f255868c7abe378ecd43dbb1adca91f753c5
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Nov 19 20:54:33 2014 -0500

    Debug Parser

commit bff270d0abdfb74cb70fa2b6a5756a9aee0758ee
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Nov 19 20:42:07 2014 -0500

    fixed list
```

```

commit 68c8327bde098c37da46ccab0e177bde4013c5c8
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Nov 19 20:41:09 2014 -0500

    edited list

commit d7ed25864869f53997fb067e46060f1bbf8e16ed
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Nov 19 20:39:05 2014 -0500

    fixed fdec

commit 641518e46dfa8eb1aeed2d0bbfa912ffd22ce882
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Nov 19 20:37:18 2014 -0500

    Parser Partial Done

commit 23bba766a45febd2f1cf92b10460a86500951228
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Nov 19 20:35:24 2014 -0500

    Test Stuff

commit e0eefa4514ac538a27fe1b96f901d78b26495949
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Wed Nov 19 19:18:30 2014 -0500

    AST update

commit 06a83d97776e13124e9a71676bbed5f424343d4d
Author: Martin Ong <mo2454@columbia.edu>
Date:   Wed Nov 19 17:41:10 2014 -0500

    Random commit

commit 74148dce3f31f7e5d0de52897cfe0d66dca9bdf6
Author: Alice Chang <avc2120@columbia.edu>
Date:   Wed Nov 19 17:37:27 2014 -0500

    Edits

commit 962f328af685ed094842ccf4a276fea2f17a31af
Author: Martin Ong <mo2454@columbia.edu>
Date:   Tue Oct 21 17:08:59 2014 -0400

    Merge parser and scanner with Martin

commit e088ac16fa73bd4891b7bec18f52f4cb70ecd9bd
```

```
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Tue Oct 21 16:56:37 2014 -0400

    Parser-Gabriel

commit b655c806a5ba804249e878a38edcbc0304e978c9
Author: detectiveconan2 <ggl2110@columbia.edu>
Date:   Tue Oct 21 16:53:09 2014 -0400

    Wrote Scanner-Gabriel

    Hi

commit 2ef66f2ca7e5ce900b0f49763680986a30c0cef8
Merge: 42222cf 3a7fcb5
Author: Martin Ong <mo2454@columbia.edu>
Date:   Tue Oct 21 16:49:44 2014 -0400

    Merge branch 'master' of https://github.com/martinong/ChemLAB

    Conflicts:
      chemlab.ml

commit 42222cf3a01bd523f1916d3fccfef3570a17853b
Author: Martin Ong <mo2454@columbia.edu>
Date:   Tue Oct 21 16:44:06 2014 -0400

    Removed implementation stuff

commit 3a7fcb5b5ba170ce7812396167a090dae519ad42
Author: Alice Chang <avc2120@columbia.edu>
Date:   Mon Oct 20 21:07:45 2014 -0400

    added print hash map variable

commit b0443e0c15fa46243a601bac1f3012ed568fcf72
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Oct 20 20:19:22 2014 -0400

    Cleaned merge mess

commit c3d8b8b80e9c2ebdf017ed3b49dc726408509fdb
Merge: 5e75369 144cfc7
Author: Alice Chang <avc2120@columbia.edu>
Date:   Mon Oct 20 20:13:44 2014 -0400

    edited

commit 5e75369a41970d398dd945a9feecaf6e65be5f9b
Author: Alice Chang <avc2120@columbia.edu>
```

```
Date:   Mon Oct 20 20:11:45 2014 -0400

    edited

commit 144cfc7f6719e62ce8fd0f521b56627c6fc7582a
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Oct 20 20:11:31 2014 -0400

    Comment working, variables are in progress

commit bb5f80d4583b4ac13ac921db8402827b6a370812
Author: Martin Ong <mo2454@columbia.edu>
Date:   Mon Oct 20 19:04:16 2014 -0400

    Print function working

    Includes test file for printing

commit bacdb7c691b8f8b52723451f771e03b05eea68a5
Merge: 846faec 865fd5e
Author: Alice Chang <avc2120@columbia.edu>
Date:   Mon Oct 20 18:06:17 2014 -0400

    Merge branch 'master' of https://github.com/martinong/ChemLAB

    Conflicts:
      scanner.mll

commit 846faecea24570fb722c87850050e93f100eebb5
Author: Alice Chang <avc2120@columbia.edu>
Date:   Mon Oct 20 18:04:42 2014 -0400

    Parser and Scanner Edited

commit 865fd5ebba8f48e28420f1ef096a428bf7b83dae
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sat Oct 11 15:29:51 2014 -0400

    Tried to add print function

commit 2c364e4b2b2c6760cb72f1574f32143a3c9d656b
Author: Alice Chang <avc2120@columbia.edu>
Date:   Sat Oct 11 15:02:25 2014 -0400

    Added tokens

commit 95b7a81b2ce176522160826f6fbf619138d0fec1
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sat Oct 11 14:51:05 2014 -0400

```

```
    Ignore files

commit c129eb2af310efc9c65b9891fdafe6c1bc333a16
Author: Alice Chang <avc2120@columbia.edu>
Date:   Sat Oct 11 14:48:52 2014 -0400

    First Edit

commit 0f4bc817af1bdcc1cdf3bb47415678ce719e73b3
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sat Oct 11 14:44:11 2014 -0400

    Updated name in makefile

commit 561e083a8896858fb1125b8fa36730f43a8d0060
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sat Oct 11 14:32:51 2014 -0400

    Changed name from calc to chemlab

commit daa59975d3beaf15a469f93c7444ed69dd9e5a1e
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sat Oct 11 14:30:04 2014 -0400

    Added variables and sequencing

    From homework 1 problem 3

commit 3ac1b97628ae7492ebb0a0b059c8c3c3838cf5ce
Author: Martin Ong <mo2454@columbia.edu>
Date:   Sat Oct 11 14:26:49 2014 -0400

    Calculator parser from COMS W4115

commit b4bff48721d629be42c09396b8e056319c08fd9e
Author: Martin Ong <martinong@users.noreply.github.com>
Date:   Sat Oct 11 13:48:27 2014 -0400

    Initial commit
```