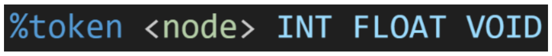
Part 2 – documentation

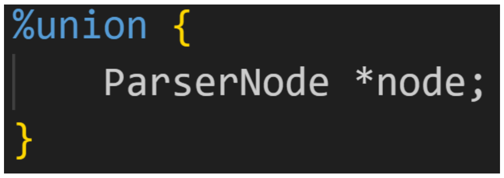
In our bison program we declared tokens which we get from our lexer.

For example:



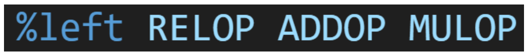
```
%token <node> INT FLOAT VOID
```

where <node> defines the type of ID which is derived from our %union implementation.



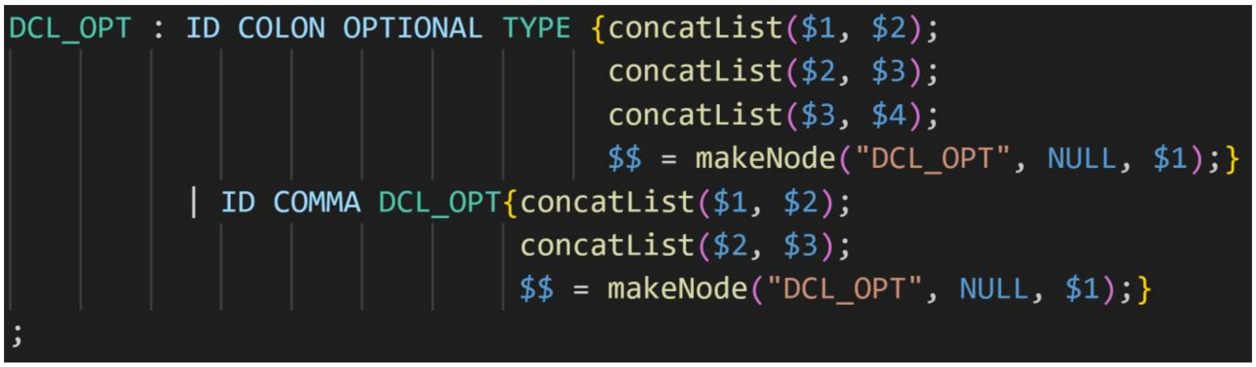
```
%union {
    ParserNode *node;
}
```

We also declared the associativity of selected tokens to represent the correctness of the code we parse.



```
%left RELOP ADDOP MULOP
```

In our grammar rules we build the tree in the following manner:



```
DCL_OPT : ID COLON OPTIONAL TYPE {concatList($1, $2);
                                  concatList($2, $3);
                                  concatList($3, $4);
                                  $$ = makeNode("DCL_OPT", NULL, $1);}
          | ID COMMA DCL_OPT{concatList($1, $2);
                             concatList($2, $3);
                             $$ = makeNode("DCL_OPT", NULL, $1);}
;
```

$$, $1, $2 and so on are node types so they represent nodes in our tree. We concatenate the symbols/terminals in the right side of the rule to each other (they become siblings) and then we connect the siblings to be the children of the parent node which is the symbols on the left side of the rule.

After parsing the whole code file, we print the tree using the provided helper function.