CS1319: Programming Language Design and Implementation
Department of Computer Science
Assignment 3: Guide

This is a guide on the output format for Assignment 3. Please read it carefully as we will be using an auto-grader. The purpose of the output format we have decided is to be able to provide you with better feedback so that we can explain your mistakes, which will hopefully mean that you can fix them before attempting the next assignment.

primary-expression:
identifier
constant
string-literal
(*expression*)

We will be using this production rule from the assignment as an example. The bison code for the above rule should look something like this in your `.y` file.

```
primary_expression: ID
                  | constant
                  | STRING_LITERAL
                  | OP_PARENTHESIS expression CL_PARENTHESIS
                  ;

constant: INTEGER_CONSTANT
        | CHAR_CONSTANT
        ;
```

It is perfectly valid to do this production without defining the non-terminal constant, but I have done so to elucidate the output format better. For a given production rule:

1. If it is present in as one of the rules in the specifications, print the **left-hand side** non-terminal.
2. If it is one of the rules you have defined yourself, print nothing.

Here, the non-terminal `primary_expression` corresponds to the non-terminal *primary-expression* in the specifications, while the non-terminal `constant` is not in the specifications.

```
primary_expression: ID {printf("primary-expression\n");}
                  | constant {printf("primary-expression\n");}
                  | STRING_LITERAL {printf("primary-expression\n");}
                  | OP_PARENTHESIS expression CL_PARENTHESIS {printf("primary-expression\n");}
                  ;

constant: INTEGER_CONSTANT
        | CHAR_CONSTANT
        ;
```

This should be how your `.y` file looks for this specific production rule. Do the same thing for every production rule present in the specifications. Note that we will not be manually fixing your output formats like last time. Be careful when printing the rule names and make sure that they exactly match what is given in the specifications.

Make sure you have a yyerror function defined to handle wrong inputs:

```
void yyerror(char *s) {
    printf("Error: %s on '%s'\n",s,yytext);
}
```