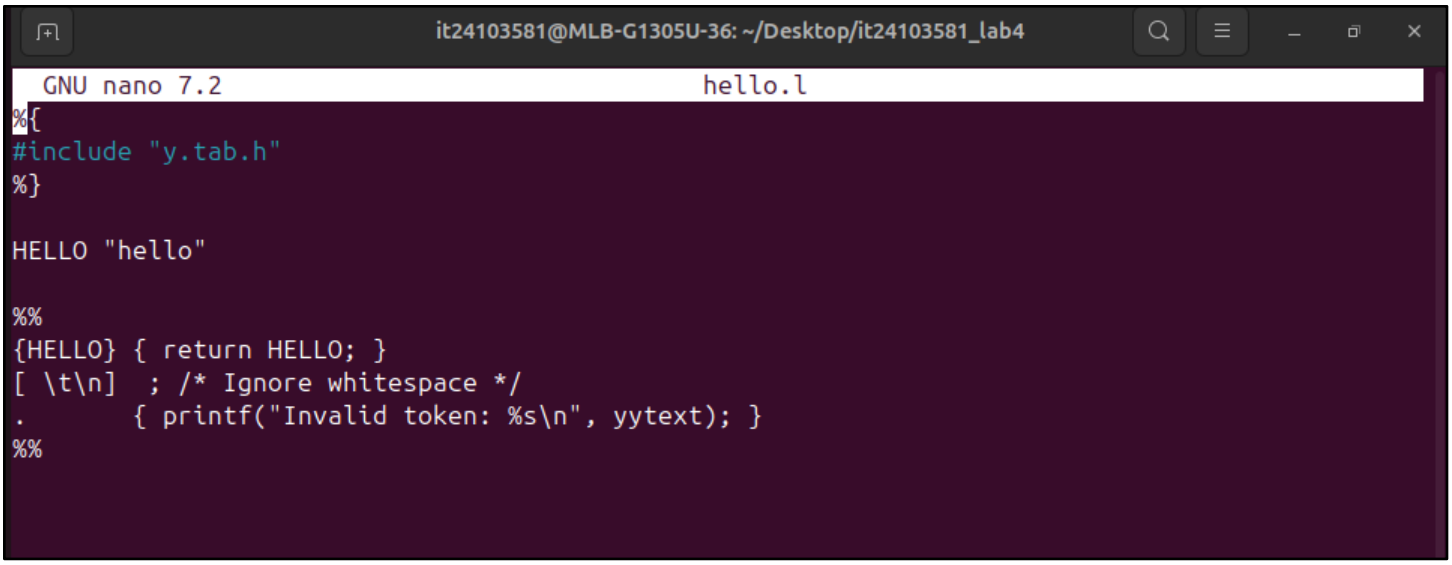
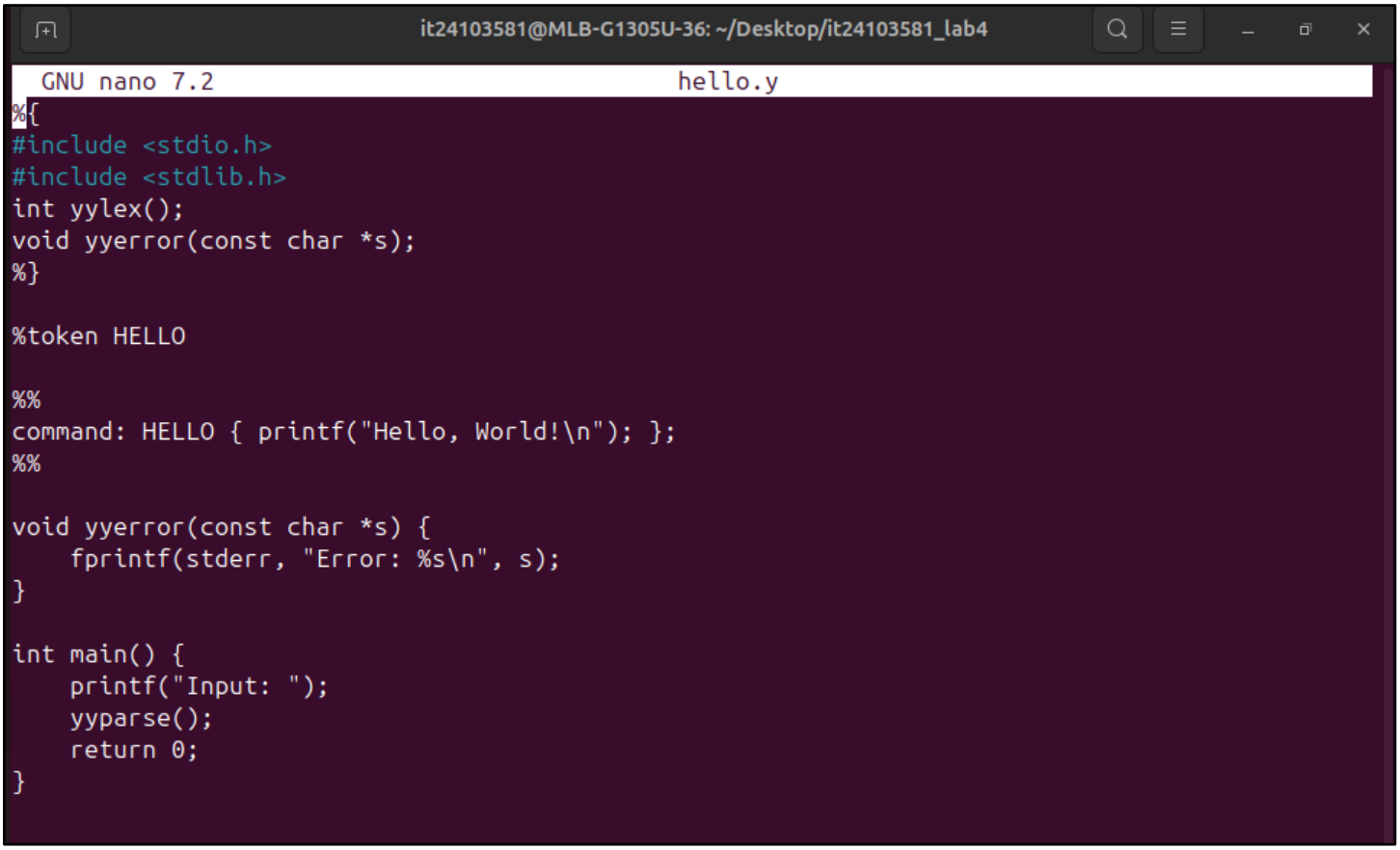
Practical 03 - Lex & YACC Implementation - IT24103581

Simple Example: Hello World Command

Lex File (hello.l)



YACC File (hello.y)



Compilation and Execution:

```
it24103581@MLB-G1305U-36:~/Desktop/it24103581_lab4$ nano hello.l
it24103581@MLB-G1305U-36:~/Desktop/it24103581_lab4$ nano hello.y
it24103581@MLB-G1305U-36:~/Desktop/it24103581_lab4$ yacc -d hello.y
it24103581@MLB-G1305U-36:~/Desktop/it24103581_lab4$ lex hello.l
it24103581@MLB-G1305U-36:~/Desktop/it24103581_lab4$ gcc -o hello y.tab.c lex.yy.c -L/mingw64/l
ib -lfl
it24103581@MLB-G1305U-36:~/Desktop/it24103581_lab4$ ./hello
Input: hello
Hello, World!
```

Implementing a Simple Calculator

BNF Grammar

<expr> ::= <expr> + <term>
        | <expr> - <term>
        | <term>

<term> ::= <term> * <factor>
        | <term> / <factor>
        | <term> % <factor>
        | <factor>

<factor> ::= NUMBER
        | (<expr>)

Lex File – calc.l

```
%{
#include "calc.tab.h"
#include <stdlib.h>
%}

%option noyywrap

%%
[0-9]+    { yylval.num = atoi(yytext); return NUMBER; }
[ \t]     ;
\n        { return '\n'; }

"+"       { return '+'; }
"-"       { return '-'; }
"*"       { return '*'; }
"/"       { return '/'; }
"%"       { return '%'; }
"("       { return '('; }
")"       { return ')'; }

.         { printf("Invalid character\n"); }
```

```
%%
```

YACC File – calc.y

```
%{
#include <stdio.h>
#include <stdlib.h>

int yylex();
void yyerror(const char *s);
%}

%union {
   int num;
}

%token <num> NUMBER
%type <num> expr

%left '+' '-'
%left '*' '/' '%'

%%

input:
    | input line
    ;

line:
    expr '\n'   { printf("Result = %d\n", $1); }
  | '\n'
    ;

expr:
    expr '+' expr   { $$ = $1 + $3; }
  | expr '-' expr   { $$ = $1 - $3; }
  | expr '*' expr   { $$ = $1 * $3; }
  | expr '/' expr
     {
        if ($3 == 0) {
           printf("Error: Division by zero\n");
           $$ = 0;
        } else {
           $$ = $1 / $3;
        }
     }
  | expr '%' expr
     {
```

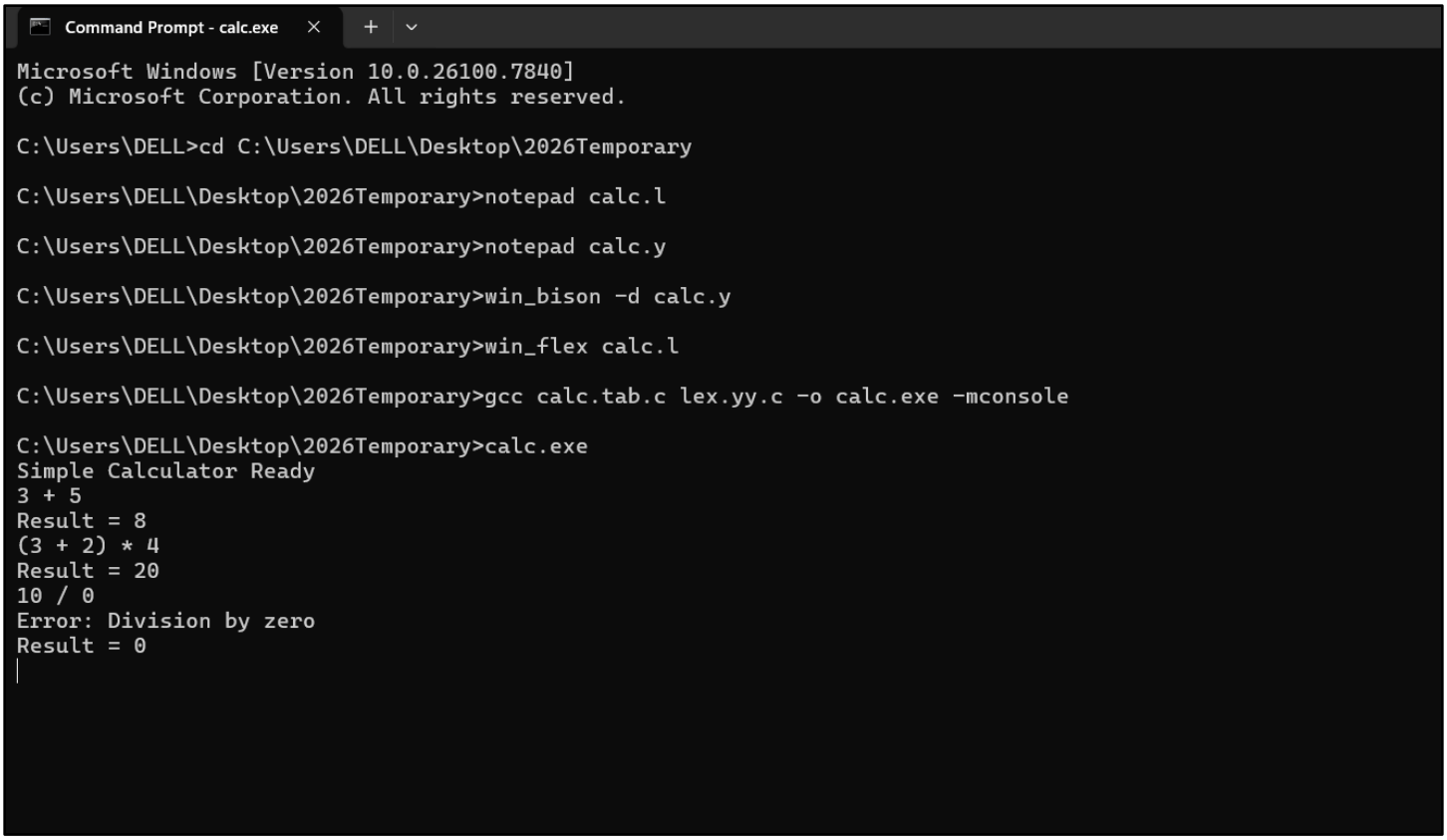
```
        if ($3 == 0) {
          printf("Error: Modulus by zero\n");
          $$ = 0;
        } else {
          $$ = $1 % $3;
        }
      }
  | '(' expr ')'   { $$ = $2; }
  | NUMBER        { $$ = $1; }
   ;

%%

int main() {
  printf("Simple Calculator Ready\n");
  yyparse();
  return 0;
}

void yyerror(const char *s) {
  printf("Syntax Error\n");
}
```

Compilation and Execution:



Section 3: Vehicle Control System

BNF Grammar

<command> ::= <component> <action>
        | <component> <action> <value>

<component> ::= engine
        | doors
        | lights
        | speed

<action> ::= start
        | stop
        | lock
        | unlock
        | on
        | off
        | set

<value> ::= NUMBER

Lex File - vehicle.l

```
%{
#include "vehicle.tab.h"
%}

%option noyywrap

%%
"engine"  { return ENGINE; }
"doors"   { return DOORS; }
"lights"  { return LIGHTS; }
"speed"   { return SPEED; }

"start"   { return START; }
"stop"    { return STOP; }
"lock"    { return LOCK; }
"unlock"  { return UNLOCK; }
"on"      { return ON; }
"off"     { return OFF; }
"set"     { return SET; }

[0-9]+    { yylval.num = atoi(yytext); return NUMBER; }

\n        { return '\n'; }
[ \t]     ;
.         { printf("Invalid input\n"); }

%%
```

Lex File - vehicle.y

```
%{
#include <stdio.h>
#include <stdlib.h>

int yylex();
void yyerror(const char *s);
%}

%union {
  int num;
  char* str;
}

%token ENGINE DOORS LIGHTS SPEED
%token START STOP LOCK UNLOCK ON OFF SET
%token <num> NUMBER

%type <str> component action

%%

input:
    | input line
    ;

line:
    command '\n'
    ;

command:
    component action
    {
      printf("Command: %s %s\n", $1, $2);
    }
  | SPEED SET NUMBER
    {
      printf("Command: Speed Set %d\n", $3);
    }
    ;

component:
    ENGINE { $$ = "Engine"; }
  | DOORS  { $$ = "Doors"; }
  | LIGHTS { $$ = "Lights"; }
  ;

action:
    START  { $$ = "Start"; }
  | STOP   { $$ = "Stop"; }
  | LOCK   { $$ = "Lock"; }
  | UNLOCK { $$ = "Unlock"; }
  | ON     { $$ = "On"; }
  | OFF    { $$ = "Off"; }
  ;

%%

int main() {
  printf("Vehicle Control System Ready\n");
  yyparse();
  return 0;
}
```

```
void yyerror(const char *s) {
   printf("Syntax Error\n");
}
```

Compilation and Execution: