```
Rhea Adhikari
190905156
Incomplete Code
Yet to implement left recursion and left factoring.
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <ctype.h>
#include "lexical.h"
struct token tok;
void declarations();
int data_type();
void identifier_list(struct token);
void assign_stat(struct token);
void prgrm();
void invalid(struct token *tok);
FILE *fptr;
void invalid(struct token *tok)
{
    printf("error at row: %d, col: %d for lexeme \" %s \" \n", tok->row, tok-
>col, tok->token_name);
    printf("invalid input error.\n");
    exit(0);
}
void valid()
{
    printf("S U C C E S S F U L\n");
    exit(0);
}
void prgrm() // to handle the first production
{
    tok = getNextToken(fptr);
    if (strcmp(tok.token_name, "main") == 0)
    {
        tok = getNextToken(fptr);
        if (strcmp(tok.token_name, "(") == 0)
        {
            tok = getNextToken(fptr);
            if (strcmp(tok.token_name, ")") == 0)
            {
                tok = getNextToken(fptr);
                if (strcmp(tok.token_name, "{") == 0)
                {
                    declarations();
                    tok = getNextToken(fptr);
```

```
                    if (strcmp(tok.token_name, "}") == 0)
                        return;
                    else
                        invalid(&tok);
                }
                else
                    invalid(&tok);
            }
            else
                invalid(&tok);
        }
        else
            invalid(&tok);
    }
    else
        invalid(&tok);
}
void declarations()
{
    tok = getNextToken(fptr);
    if (data_type(tok.token_name))
    {
        tok = getNextToken(fptr);
        identifier_list(tok);
        tok = getNextToken(fptr);
        if (strcmp(tok.token_name, ";") == 0)
            declarations();
        else
            invalid(&tok);
    }
    else
        assign_stat(tok);
}
int data_type(char *ch)
{
    if (strcmp(ch, "int") == 0 || strcmp(ch, "char") == 0)
        return 1;
    else
        return 0;
}
void identifier_list(struct token tkn)
{
    struct token tok1;
    tok1 = tok;
    if (strcmp(tok1.token_name, "id") == 0)
```

```
    {
        tok1 = getNextToken(fptr);
        if (strcmp(tok1.token_name, ",") == 0)
        {
            tok1 = getNextToken(fptr);
            identifier_list(tok);
        }
        else if (strcmp(tok1.token_name, ";") == 0)
        {
            fseek(fptr, -1, SEEK_CUR);
            return;
        }
        else
            invalid(&tok1);
    }
}
void assign_stat(struct token tok)
{
    struct token tok1;
    if (strcmp(tok.token_name, "id") == 0)
    {
        tok1 = getNextToken(fptr);
        if (strcmp(tok1.token_name, "=") == 0)
        {
            tok1 = getNextToken(fptr);
            if (strcmp(tok1.token_name, "num") == 0 || strcmp(tok1.token_name,
"id"))
            {
                tok1 = getNextToken(fptr);
                if (strcmp(tok1.token_name, ";") == 0)
                    return;
                else
                    invalid(&tok);
            }
            else
                invalid(&tok);
        }
        else
            invalid(&tok);
    }
}
int main()
{
    fptr = fopen("sample.c", "r");
    if (fptr == NULL)
```

```
    {
        printf("File cannot be opened!\n");
        return 0;
    }
    prgrm();
    valid();
}
```