**COMPILER DESIGN WEEK 7**

Q1. For given subset of grammar 7.1, design RD parser with appropriate error messages with expected character and row and column number.

PROGRAM:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

#include "lexanls.h"

void program();

void declarations();

void data\_type();

void identifier\_list();

void assign\_stat();

struct token curr;

FILE \*f1;

void invalid()

{

printf("error");

exit(0);

}

//PROGRAM FUNCTION

void program()

{

curr=getNextToken(f1);

if(isdtype(curr.lexeme))

{

curr=getNextToken(f1);

if(strcmp(curr.lexeme,"main")==0)

{

curr=getNextToken(f1);

if(strcmp(curr.lexeme,"(")==0)

{

curr=getNextToken(f1);

if(strcmp(curr.lexeme,")")==0)

{

curr=getNextToken(f1);

if(strcmp(curr.lexeme,"{")==0)

{

curr=getNextToken(f1);

declarations();

assign\_stat();

if(strcmp(curr.lexeme,"}")==0)

{

return;

}

else

{

printf("\n missing } at row:%d and col:%d.\n",curr.row,curr.col);

exit(0);

}

}

else

{

printf("\n missing { at row:%d and col:%d.\n",curr.row,curr.col);

exit(0);

}

}

else

{

printf("\n missing ) at row:%d and col:%d.\n",curr.row,curr.col);

exit(0);

}

}

else

{

printf("\n missing ( at row:%d and col:%d.\n",curr.row,curr.col);

exit(0);

}

}

else

{

printf("\n missing main function\n");

exit(0);

}

}

else

{

printf("\n missing return datatype of function\n");

exit(0);

}

}

//DECLARATION FUNCTION

void declarations()

{

if(isdtype(curr.lexeme))

{

data\_type();

identifier\_list();

if(strcmp(curr.lexeme,";")==0)

{

curr=getNextToken(f1);

declarations();

}

else printf("\n missing ; at row:%d and col:%d.\n",curr.row,curr.col);

}

}

//DATA TYPE FUNCTION

void data\_type()

{

if(strcmp(curr.lexeme,"int")==0)

{

curr=getNextToken(f1);

return;

}

else if(strcmp(curr.lexeme,"char")==0)

{

curr=getNextToken(f1);

return;

}

else

{

printf("\n missing data type at row:%d and col:%d.\n",curr.row,curr.col);

exit(0);

}

}

//INDENTIFIER FUNCTION

void identifier\_list()

{

if(strcmp(curr.type,"IDENTIFIER")==0)

{

curr=getNextToken(f1);

if(strcmp(curr.lexeme,",")==0)

{

curr=getNextToken(f1);

identifier\_list();

}

else return;

}

else

{

printf("\n missing identifier at row:%d and col:%d.\n",curr.row,curr.col);

exit(0);

}

}

//ASSIGN FUNCTION

void assign\_stat()

{

if(strcmp(curr.type,"IDENTIFIER")==0)

{

curr=getNextToken(f1);

if(strcmp(curr.lexeme,"=")==0)

{

curr=getNextToken(f1);

if(strcmp(curr.type,"IDENTIFIER")==0)

{

curr=getNextToken(f1);

if(strcmp(curr.lexeme,";")==0)

{

curr=getNextToken(f1);

return;

}

}

else if(strcmp(curr.type,"NUMBER")==0)

{

curr=getNextToken(f1);

if(strcmp(curr.lexeme,";")==0)

{

curr=getNextToken(f1);

return;

}

else

{

printf("\n missing ; at row:%d and col:%d.\n",curr.row,curr.col);

exit(0);

}

}

else

{

printf("\n missing identifier at row:%d and col:%d.\n",curr.row,curr.col);

exit(0);

}

}

else

{

printf("\n missing = at row:%d and col:%d.\n",curr.row,curr.col);

exit(0);

}

}

else

{

printf("\n missing identifier at row:%d and col:%d.\n",curr.row,curr.col);

exit(0);

}

}

//MAIN FUNCTION

int main()

{

FILE \*fa, \*fb;

int ca, cb;

fa = fopen("inp\_71.c", "r");

if (fa == NULL)

{

printf("Invalid file\n");

return 0;

}

fb = fopen("out\_71.c", "w");

ca = getc(fa);

while (ca != EOF)

{

if(ca==' ')

{

putc(ca,fb);

while(ca==' ')

ca = getc(fa);

}

if (ca=='/')

{

cb = getc(fa);

if (cb == '/')

{

while(ca != '\n')

ca = getc(fa);

}

else if (cb == '\*')

{

do

{

while(ca != '\*')

ca = getc(fa);

ca = getc(fa);

} while (ca != '/');

}

else

{

putc(ca,fb);

putc(cb,fb);

}

}

else putc(ca,fb);

ca = getc(fa);

}

fclose(fa);

fclose(fb);

fa = fopen("out\_71.c", "r");

if(fa == NULL)

{

printf("Invalid file");

return 0;

}

fb = fopen("temp.c", "w");

ca = getc(fa);

while(ca != EOF)

{

if(ca == '#')

{

while(ca != '\n')

{

ca = getc(fa);

}

}

ca = getc(fa);

if(ca != EOF && ca != '#')

{

putc(ca, fb);

}

}

fclose(fa);

fclose(fb);

fa = fopen("temp.c", "r");

fb = fopen("out\_71.c", "w");

ca = getc(fa);

while(ca != EOF)

{

putc(ca, fb);

ca = getc(fa);

}

fclose(fa);

fclose(fb);

remove("temp.c");

f1=fopen("out\_71.c","r");

if(f1==NULL)

{

printf("Invalid file\n");

return 0;

}

struct token tkn;

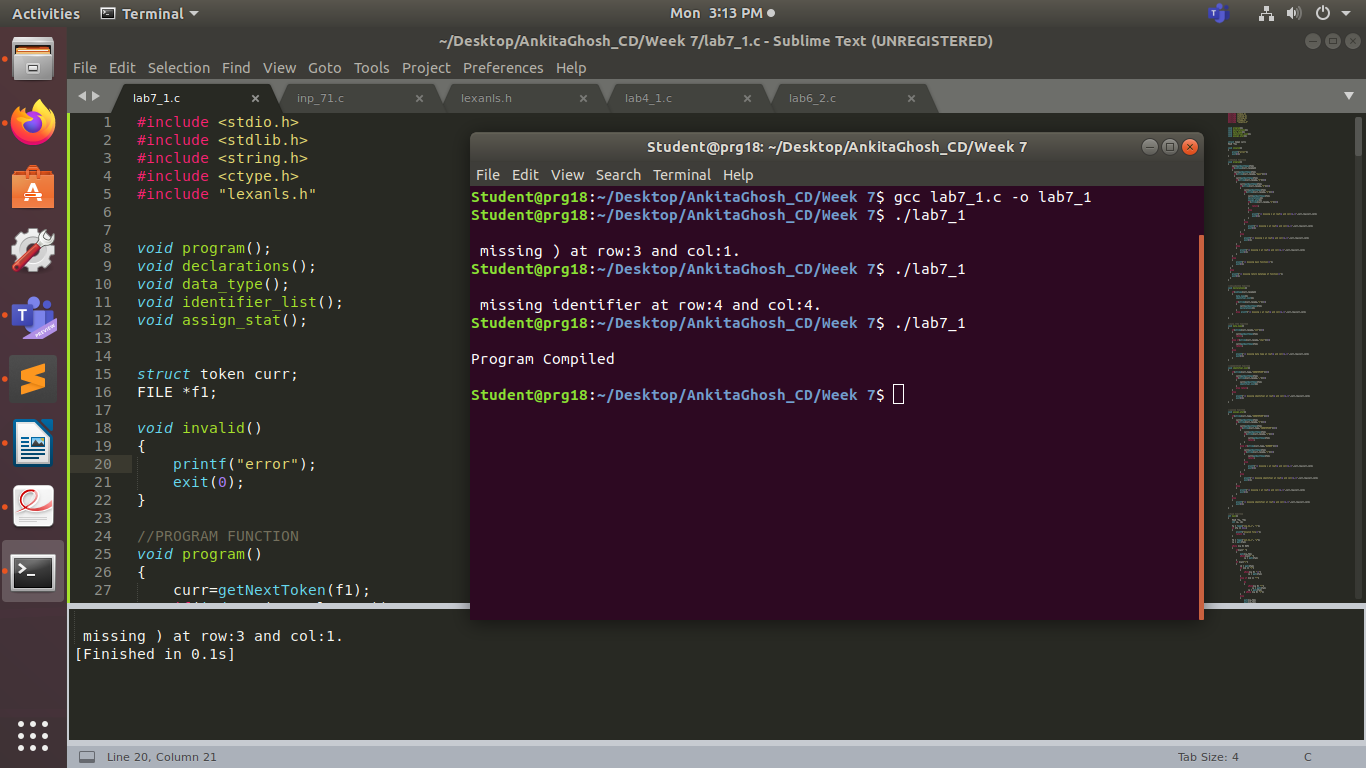
program();

printf("\nProgram Compiled\n\n");

fclose(f1);

}

OUTPUT:



*Input Program in 1st run:*

#include <stdio.h>

int main(

{ int a,b,c; char d;

a=1;

}

*Input Program in 2nd run:*

#include <stdio.h>

int main()

{ int a,b,c; char d;

a=;

}

*Input Program in 3rd run:*

#include <stdio.h>

int main()

{ int a,b,c; char d;

a=1;

}