**Practical – 10**

**Aim:**

**(a) Write a YACC program to validate syntax of function prototype.**

**(b) Write a YACC program to validate nested IF control statements.**

**Code (a):**

**practical10a.l**

%{

#include "y.tab.h"

%}

VOID "void"|"void "

TYPE "int "|"bool "|"char "|"double "

LETTER [a-zA-Z\_]

DIGIT [0-9]

%%

{VOID} return VOID;

{TYPE} return TYPE;

{LETTER}({LETTER}|{DIGIT})\* return ID;

"(" return LPARAN;

")" return RPARAN;

"," return COMMA;

";" return SEMI;

"\*" return POINTER;

"&" return ADDRESS;

[ \t] /\* ignore white space \*/;

"\n" return 0;

. return yytext[0];

%%

**practical10a.y**

%{

#include <stdio.h>

void yyerror(char \*s);

int yylex(void);

%}

%token VOID TYPE LPARAN RPARAN COMMA SEMI ID POINTER ADDRESS

%%

START: S {

printf("function prototype is valid!\n");

return 0;

}

S: RTYPE FNAME LPARAN INNER RPARAN SEMI

;

INNER: TYPE IDENTIFIER

|

TYPE IDENTIFIER COMMA INNER

|

VOID

|

;

FNAME: ID

|

POINTER ID

;

IDENTIFIER: ID

|

POINTER ID

|

ADDRESS ID

;

RTYPE: TYPE

|

VOID

;

%%

void yyerror(char \*s)

{

fprintf(stderr, "Invalid function prototype (%s: detected)\n", s);

}

int main()

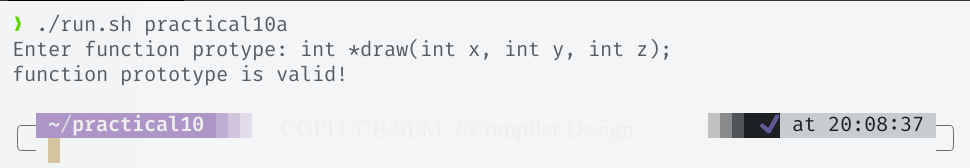
{

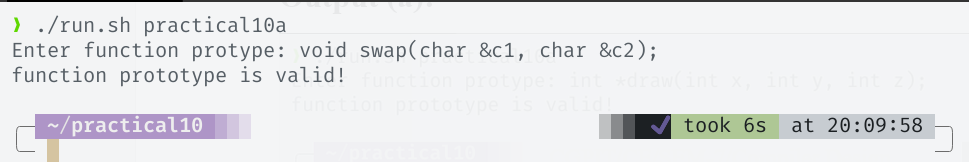
printf("Enter function protype: ");

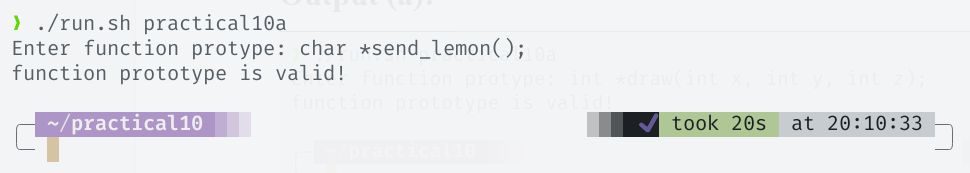
return yyparse();

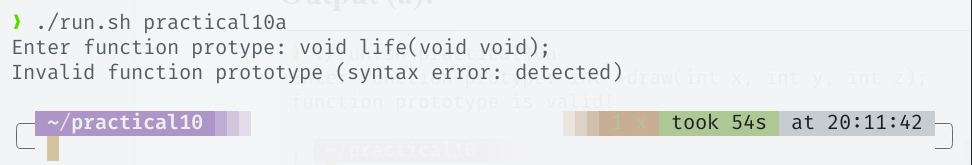
}

**Output (a):**

****

****

****

****

**Code (b):**

**practical10b.l**

%{

#include "y.tab.h"

%}

LETTER [a-zA-Z\_]

DIGIT [0-9]

OPR =|<|>|!=|<=|>=|==|&&|"||"|[+\-\*/]

%%

"if" return IF;

"else" return ELSE;

{DIGIT}+ return NUMBER;

{LETTER}({LETTER}|{DIGIT})\* return ID;

{OPR} return OPR;

"(" return LPARAN;

")" return RPARAN;

"{" return LBRACE;

"}" return RBRACE;

";" return SEMICOLON;

"," return COMMA;

"++"|"--" return UOP;

[ \t] /\* ignore white space \*/;

"\n" return 0;

. /\* ignore everything else \*/;

%%

**practical10b.y**

%{

#include <stdio.h>

void yyerror(char \*s);

int yylex(void);

%}

%token IF ELSE LPARAN RPARAN ID NUMBER LBRACE RBRACE SEMICOLON UOP OPR COMMA

%%

S1: S {

printf("conditional statement is valid!\n");

return 0;

};

S: IFCONDITION

;

IFCONDITION: IF LPARAN CEXP RPARAN CONTINUE

;

CONTINUE: IFCONDITION /\* for nested if condition \*/

| CBODY CONTINUE /\* for statements inside control statements \*/

| LBRACE IFCONDITION RBRACE /\* case where braces were used for if blocks \*/

| ELSE CBODY CONTINUE /\* for else statements \*/

|

;

CBODY: BLOCK /\* for multiple statements in control block \*/

| STMT /\* for single statement in control block \*/

;

BLOCK: LBRACE STMTLIST RBRACE

;

STMTLIST: STMTLIST STMT

|

;

STMT: CEXP SEMICOLON

;

CEXP: EXP

| LPARAN CEXP RPARAN /\* for expressions in parenthesis (e.g. (a+b)) \*/

| CEXP COMMA EXP /\* for multiple expressions in control expression (e.g. a++, b++)\*/

|

;

EXP: ID

| NUMBER

| UOP ID /\* for unary operators (e.g. ++i)\*/

| ID UOP /\* for unary operators (e.g. i++)\*/

| ID NUMBER /\* for return number cases (e.g. return 0)\*/

| ID OPR EXP /\* for other operators (e.g. i + 1)\*/

| NUMBER OPR EXP /\* for number expressions (e.g. exp && exp)\*/

;

%%

void yyerror(char \*s)

{

fprintf(stderr, "Invalid control statement (%s: detected)\n", s);

}

int main()

{

printf("Enter control statement: ");

return yyparse();

}

**Output (b):**

