NAME: IMON RAJ

ROLL: 002010501098

CLASS: BCSE-III

SESSION: 2020-‘24

SUBJECT: COMPILER DESIGN ASSIGNMENT 3

SECTION: A3

MATCHING IF-ELSE CONSTRUCT:

LEX FILE(TO GENERATE TOKENS):

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


%%
"if"         { return IF; }
"else"       { return ELSE; }
"("          { return LPAREN; }
")"          { return RPAREN; }
"{"          { return LBRACE; }
"}"          { return RBRACE; }
";"          { return SEMICOLON; }
[0-9]+       { yylval.num = atoi(yytext); return NUMBER; }
[a-zA-Z]+    { yylval.name = strdup(yytext); return NAME; }
"=="         { return EQ; }
"="         { return ASSIGN; }
"*"         { return MULT; }
"/"         { return SLASH; }
"+"         { return PLUS; }
"-"         { return MINUS; }
"$"         { return DELIMITER; }
"!="         { return NEQ; }
">"          { return GT; }
">="         { return GTE; }
"<"          { return LT; }
"<="         { return LTE; }
```

```
"&&"        { return AND; }
"||"        { return OR; }
[ \t\n]     { /* ignore whitespace */ }
.           { }
%%

int yywrap() {
    return 1;
}
```

YACC FILE (TO CHECK SYNTAX USING CFG's):

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

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

%}

%union {
    char *name;
    int num;
}

%token <name> NAME
%token <num> NUMBER
%token IF ELSE
%token EQ NEQ GT GTE LT LTE AND OR ASSIGN PLUS MULT SLASH
MINUS DELIMITER
%token LPAREN RPAREN LBRACE RBRACE SEMICOLON

%left OR
%left AND
%left EQ NEQ
%left GT GTE LT LTE
%left PLUS MINUS
%left MULT SLASH
%left ASSIGN DELIMITER


%%
```

```
program:  statements DELIMITER { printf("\nA PROPER IF-ELSE
SYNTAX IS MATCHED...."); }
        ;

statements: statements statement | ;

statement: expr SEMICOLON
         | NAME ASSIGN expr SEMICOLON
         | IF LPAREN expr RPAREN LBRACE  statements RBRACE
         | IF LPAREN expr RPAREN LBRACE  statements RBRACE
ELSE LBRACE  statements RBRACE
         ;


expr: NUMBER
    | NAME
    | expr PLUS expr
    | expr MINUS expr
    | expr MULT expr
    | expr SLASH expr
    | expr EQ expr
    | expr NEQ expr
    | expr GT expr
    | expr GTE expr
    | expr LT expr
    | expr LTE expr
    | expr AND expr
    | expr OR expr
    | LPAREN expr RPAREN
    ;

%%
void yyerror(const char *s) {
        fprintf(stderr, "%s - IT IS NOT A PROPER IF-ELSE
CONSTRUCT\n", s);
}


int main() {
    yyparse();
    return 0;
}
```

MATCHING FOR-LOOP CONSTRUCT:

LEX FILE(TO GENERATE TOKENS):

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

%%
"for"        { return FOR; }
"("          { return OPEN_PAREN; }
";"          { return SEMICOLON; }
")"          { return CLOSE_PAREN; }
"+"          { return PLUS; }
"-"          { return MINUS; }
"*"          { return TIMES; }
"/"          { return DIVIDE; }
"{"          { return OPEN_BRACE; }
"}"          { return CLOSE_BRACE; }
"="          { return ASSIGN; }
"=="         { return EQ; }
"$"          { return DELIM; }
"!="         { return NEQ; }
">"          { return GT; }
">="         { return GTE; }
"<"          { return LT; }
"<="         { return LTE; }
"&&"         { return AND; }
"||"         { return OR; }
[0-9]+               {  return NUMBER; }
[a-zA-Z][a-zA-Z0-9]* { return NAME; }
[ \t\n]              { /* Ignore whitespace */ }
.                    { /* Ignore anything else */ }

%%

int yywrap() {
    return 1;
}
```

YACC FILE (TO CHECK SYNTAX USING CFG's):

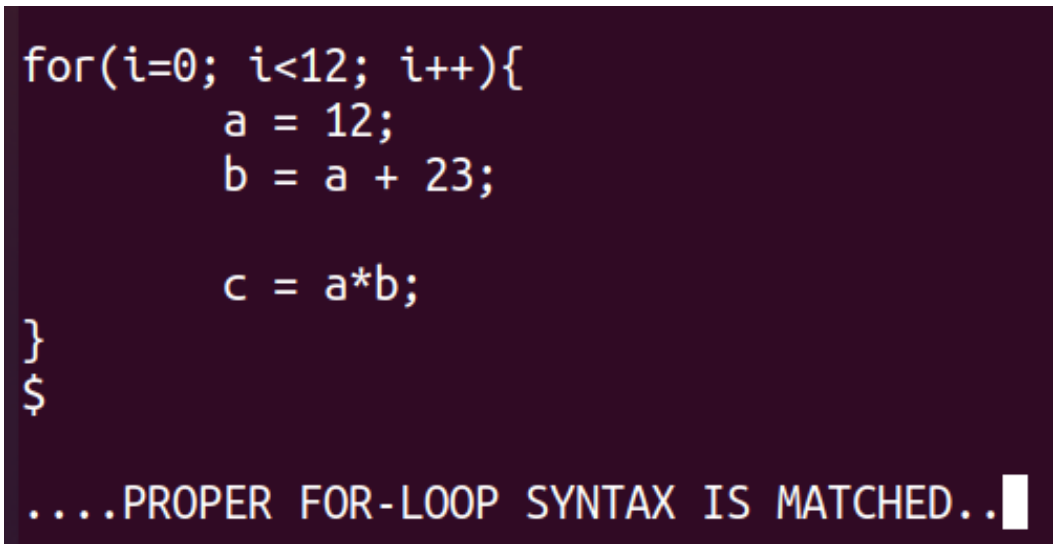
```
%{
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

extern int yylex();
extern int yylineno;
extern char* yytext;
void yyerror(const char*);
```

```
%}

%token NAME NUMBER DELIM FOR OPEN_PAREN CLOSE_PAREN SEMICOLON
PLUS ASSIGN EQ NEQ GT GTE LT LTE AND OR MINUS TIMES DIVIDE
OPEN_BRACE CLOSE_BRACE
%left NAME NUMBER DELIM FOR OPEN_PAREN CLOSE_PAREN SEMICOLON
PLUS ASSIGN EQ NEQ GT GTE LT LTE AND OR MINUS TIMES DIVIDE
OPEN_BRACE CLOSE_BRACE

%%


for_loop:
    FOR OPEN_PAREN start SEMICOLON condition SEMICOLON update
CLOSE_PAREN  OPEN_BRACE
    statements CLOSE_BRACE DELIM { printf("\n....PROPER FOR-
LOOP SYNTAX IS MATCHED.."); }
    ;

statements: statements statement | ;

statement: expr SEMICOLON
         | NAME ASSIGN expr SEMICOLON
         ;


expr: NUMBER
    | NAME
    | expr PLUS expr
    | expr MINUS expr
    | expr TIMES expr
    | expr DIVIDE expr
    | OPEN_PAREN expr CLOSE_PAREN
    ;

start:
    NAME ASSIGN NUMBER |
    ;


condition:
    boolexpr
    |
    ;

boolexpr: expr EQ expr
```

```
    | expr NEQ expr
    | expr GT expr
    | expr GTE expr
    | expr LT expr
    | expr LTE expr
    | expr AND expr
    | expr OR expr
    ;

update:
    NAME PLUS PLUS
    | NAME MINUS MINUS
    |
    ;

%%

void yyerror(const char* s) {
    fprintf(stderr, "Line %d: %s near token %s\n", yylineno,
s, yytext);
}

int main() {
    yyparse();
    return 0;
}
```

OUTPUT:

'$' is used as delimiter here..



```
if (value>=10){
        value = value * 2;
}else{
        value = value / 3;
}
$

A PROPER IF-ELSE SYNTAX IS MATCHED....
```