EXP NO: **DATE:**

RECOGNIZE A VALID CONTROL STRUCTURES SYNTAX OF C LANGUAGE (FOR LOOP, WHILE LOOP, IF-ELSE, IF-ELSE-IF, SWITCH CASE, ETC.,

AIM:

To design and implement a LEX and YACC program that recognizes the syntax of common control structures in C programming, including:

For loop

- While loop
- If-else
- If-else-if
- Switch-case

ALGORITHM:

LEX (Lexical Analyzer)

1. Start
2. Define token patterns for:
    - Keywords (e.g., if, else, for, while, switch, case)
    - Identifiers (variable names)
    - Operators (arithmetic and relational)
    - Parentheses (), {}, etc.)
    - Semicolon (;)
3. Pass recognized tokens to YACC for syntax validation.
4. End

YACC (Syntax Analyzer)

1. Start
2. Define grammar rules for:
    - For loop: for(initialization; condition; increment) { ... }
    - While loop: while(condition) { ... }
    - If-else: if(condition) { ... } else { ... }
    - If-else-if: if(condition) { ... } else if(condition) { ... } else { ... }
    - Switch-case: switch(expression) { case value: ... default: ... }
3. Parse the input expression and validate the syntax of the control structures.
4. Print appropriate messages for valid or invalid control structure syntax.
5. End

PROGRAM:

LEX File (control_structures.l):

```
%{
#include "y.tab.h"
%}
%%
"if"        { return IF; }
"else"      { return ELSE; }
"for"       { return FOR; }
"while"     { return WHILE; }
"switch"    { return SWITCH; }
"case"      { return CASE; }
[a-zA-Z_][a-zA-Z0-9_]* { return IDENTIFIER; }
"=="|"!="|"<="|">="|"<"|">" { return REL_OP; }
"+"|"-"|"*"|"/" { return ARITH_OP; }
"("         { return LPAREN; }
")"         { return RPAREN; }
"{"         { return LBRACE; }
"}"         { return RBRACE; }
";"         { return SEMICOLON; }
[ \t\n]     ; /* Ignore whitespace */
.           { printf("Invalid character: %s\n", yytext); }
%%
int yywrap() {
    return 1;
}
```

YACC File (control_structures.y)

```
%{
#include <stdio.h>
#include <stdlib.h>
void yyerror(const char *s);
int yylex(void);
%}
%token IF ELSE FOR WHILE SWITCH CASE IDENTIFIER REL_OP ARITH_OP
%token LPAREN RPAREN LBRACE RBRACE SEMICOLON
%start program
%%
program:
    statement
  | program statement
  ;
statement:
    if_statement
  | for_loop
  | while_loop
  | switch_case
  ;
```

```
if_statement:
    IF LPAREN condition RPAREN LBRACE statements RBRACE
  | IF LPAREN condition RPAREN LBRACE statements RBRACE ELSE LBRACE
statements RBRACE
  ;
for_loop:
    FOR LPAREN assignment SEMICOLON condition SEMICOLON assignment RPAREN
LBRACE statements RBRACE
  ;
while_loop:
    WHILE LPAREN condition RPAREN LBRACE statements RBRACE
  ;

switch_case:
    SWITCH LPAREN expression RPAREN LBRACE case_statements RBRACE
  ;
case_statements:
    CASE expression COLON statements
  | case_statements CASE expression COLON statements
  | case_statements DEFAULT COLON statements
  ;
condition:
    IDENTIFIER REL_OP IDENTIFIER
  | IDENTIFIER REL_OP NUMBER
  | NUMBER REL_OP IDENTIFIER
  | NUMBER REL_OP NUMBER
  ;
assignment:
    IDENTIFIER '=' expression
  ;
expression:
    IDENTIFIER
  | NUMBER
  | expression ARITH_OP expression
  ;
statements:
    statement
  | statements statement
  ;
%%
void yyerror(const char *s) {
    fprintf(stderr, "Error: %s\n", s);
}

int main() {
    printf("Enter C control structures for validation:\n");
    yyparse();
    return 0;
}
```

OUTPUT :

```
yacc -d control_structures.y
lex control_structures.l
gcc lex.yy.c y.tab.c -o control_validator
./control_validator
```

```
if (a > b) {
   // statements
} else {
   // statements
}
```

```
for (int i = 0 i < 10; i++) {
   // statements
}
```

Implementation	
Output/Signature	

RESULT:

Thus the above program to recognize a valid control structures syntax of c language (for loop, while loop, if-else, if-else-if, switch case as been implemented and executed successfully with LEX and YACC.