Practical 6

Name: Tushar Harsora

Roll Number: 19BCE509

# Explanation

There are 2 files for generating 3 address codes. The first one is parser which generates the Token from input stream. And second one is syntax analyser this file checks syntax rules and along with syntax rules it will emit the 3 address codes. Both files are attached below

# Code

practical.l

|  |
| --- |
| %{      #include "y.tab.h"  %}  %%  [0-9]+  |  [0-9]\*.[0-9]+           { yylval.value = atof(yytext); return CONSTANT; }  [a-zA-Z][a-zA-Z0-9]\*    { yylval.name = strdup(yytext); return VARIABLE; }  ";"                     { return SEPERATOR; }  [ \n\t]+                {  }  .                       { return yytext[0]; }  %% |

practical.y

|  |
| --- |
| %{      #include "y.tab.h"      #include <stdio.h>      char\* make\_temp\_name();      int yylex();      void yyerror(char\*);  %}  %union{      char\* name;      float value;  }  // %define parse.error detailed  %right '='  %left '+''-'  %left '\*''/'  %token <value> CONSTANT  %token <name> VARIABLE  %token SEPERATOR  %type <name> E  %type S  %%  S : E SEPERATOR          {              printf("ANSWER = %s\n", $1); fflush(stdout);          }  E : E '+' E          {              $$ = make\_temp\_name();              printf("%s = %s + %s\n", $$, $1, $3); fflush(stdout);          }      | E '-' E          {              $$ = make\_temp\_name();              printf("%s = %s - %s\n", $$, $1, $3); fflush(stdout);          }      | E '\*' E          {              $$ = make\_temp\_name();              printf("%s = %s \* %s\n", $$, $1, $3); fflush(stdout);          }      | E '/' E          {              $$ = make\_temp\_name();              printf("%s = %s / %s\n", $$, $1, $3); fflush(stdout);          }      | CONSTANT          {              int required = snprintf(NULL, 0, "%f", $1);              char\* buff = (char\*) malloc(required + 1);              snprintf(buff, required + 1, "%f", $1);              $$ = buff;          }      | VARIABLE          {              $$ = $1;          }      | '('E')'          {              $$ = $2;          }      | '-'E          {              printf("%s = -%s\n", $$, $2); fflush(stdout);              $$ = make\_temp\_name();          }      | E'='E          {              printf("%s = %s\n", $$, $3); fflush(stdout);              $$ = $3;          }  %%  char\* make\_temp\_name(){      static int counter = 0;      char\* name = malloc(sizeof(char) \* 10);      sprintf(name, "temp%d", counter);      counter++;      return name;  }  void yyerror(char\* s) {      printf("ERROR: %s\n", s);  }  int main(){      yyparse();      return 0;  } |

# Screenshots



