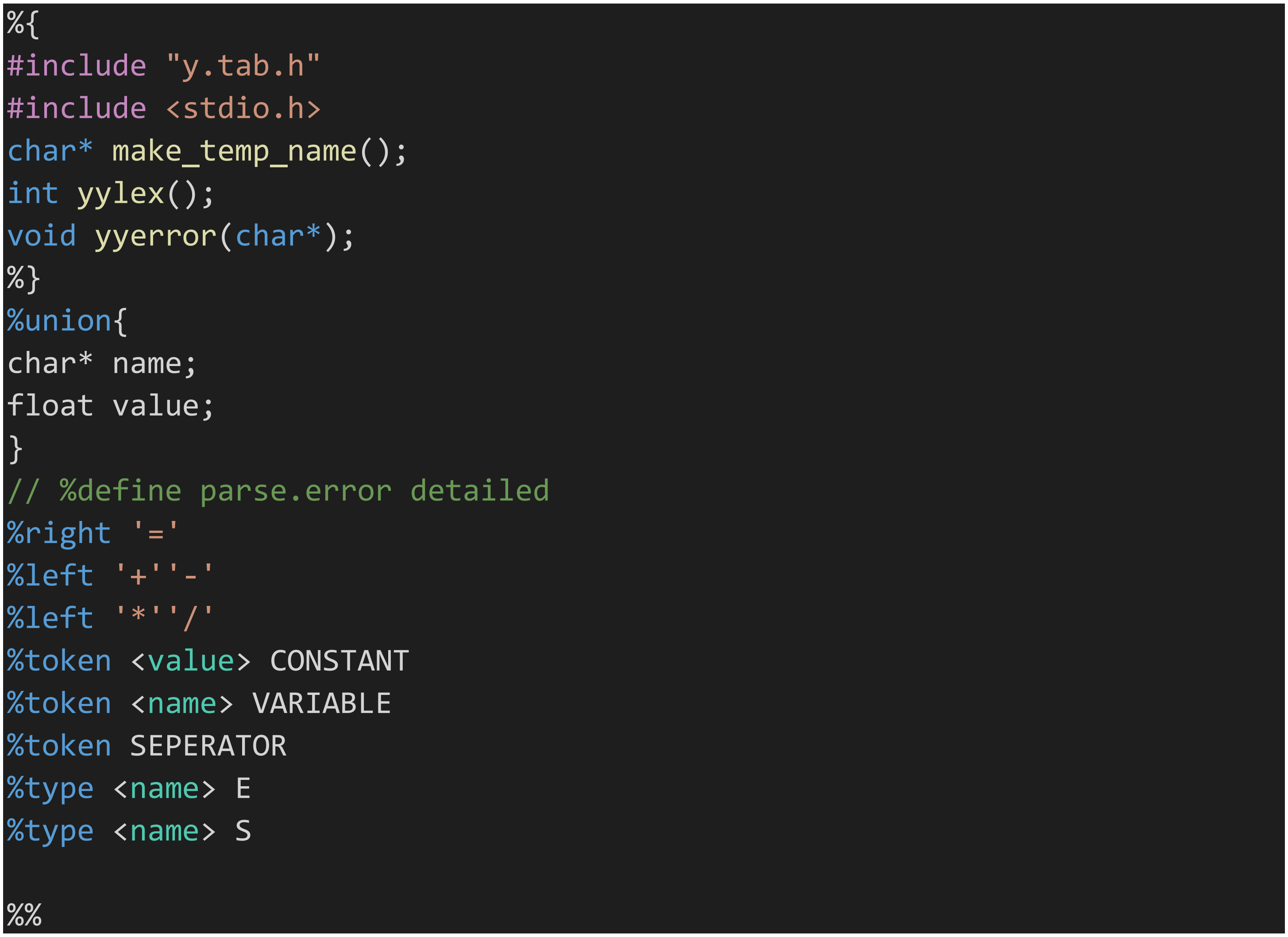
Practical No: 6

Name : Patel Savankumar P.
Enroll No : 19BCE519
Subject : Compiler Construction

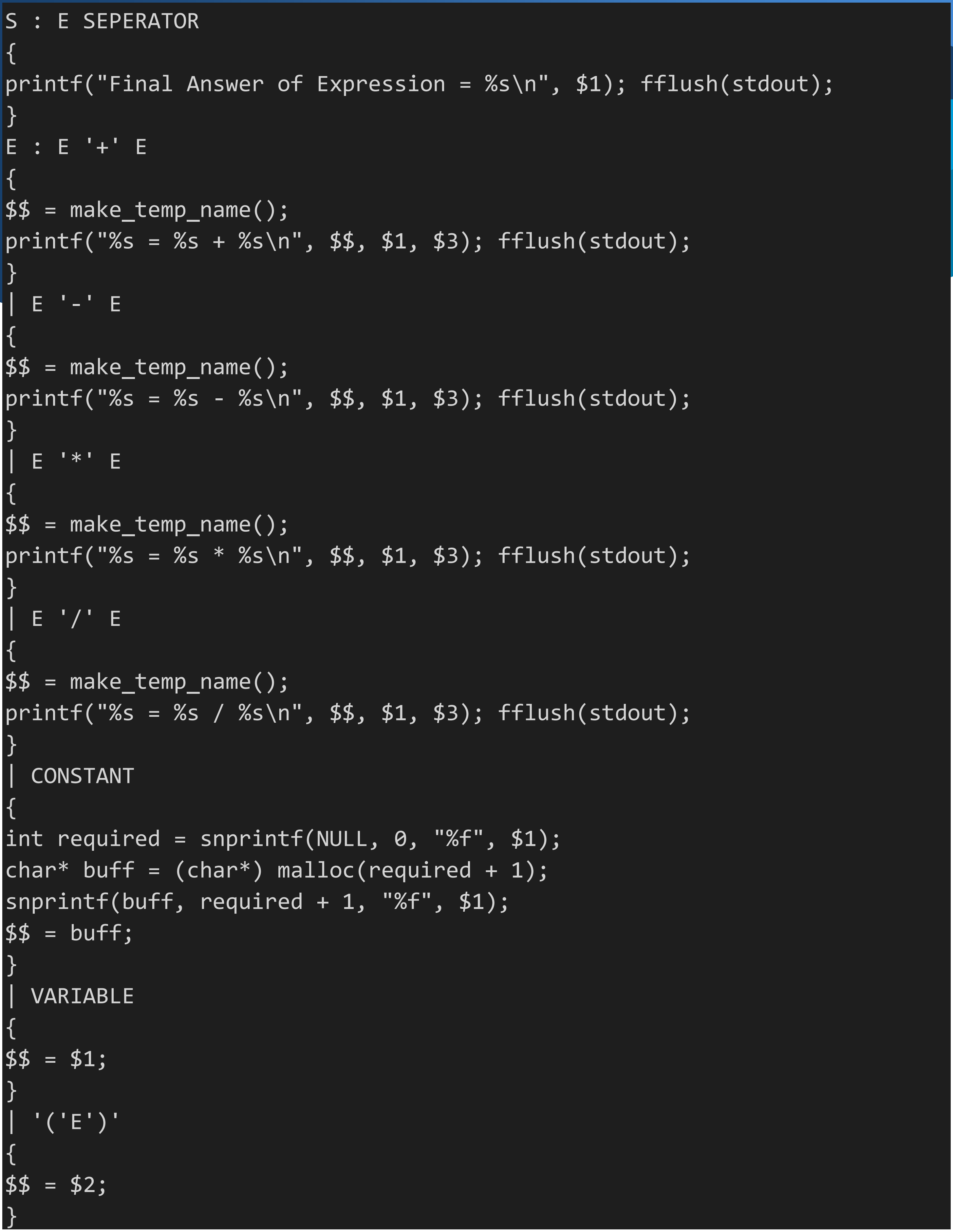
AIM: To generate 3 Address Code for Assignment.

File 1: Practical_6_Yacc.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 <name> S

%%
```

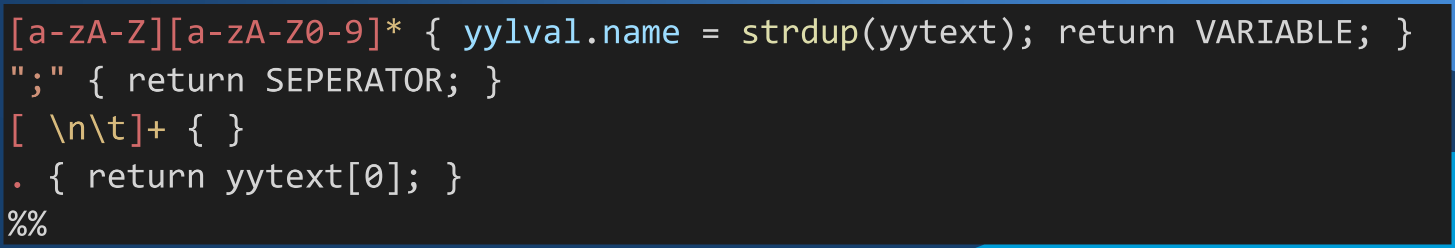


```
S : E SEPERATOR
{
printf("Final Answer of Expression = %s\n", $1); fflush(stdout);
}
E : E '+' E
{
$$ = make_temp_name();
printf("%s = %s + %s\n", $$, $1, $3); fflush(stdout);
}
| E '-' E
{
$$ = make_temp_name();
printf("%s = %s - %s\n", $$, $1, $3); fflush(stdout);
}
| E '*' E
{
$$ = make_temp_name();
printf("%s = %s * %s\n", $$, $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;
}
```

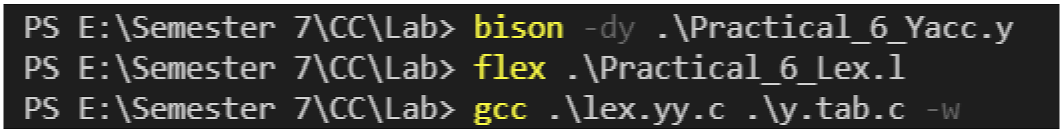
File 2: Practical_6_Lex.l

```
%option noyywrap
%{
#include "y.tab.h"
%}
%%
[0-9]+ |
[0-9]*.[0-9]+ { yylval.value = atof(yytext); return CONSTANT; }
```



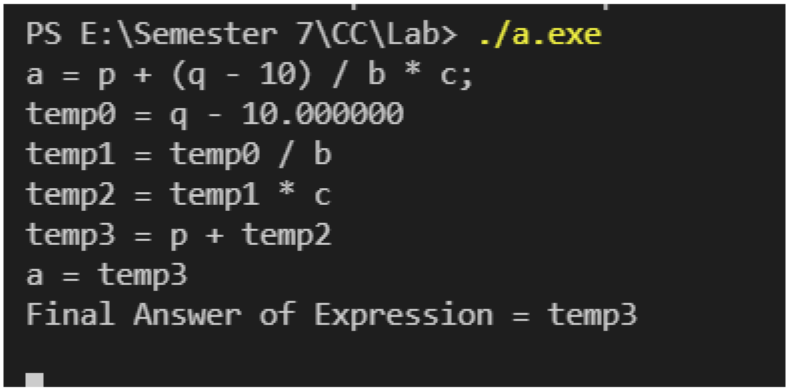
```
[a-zA-Z][a-zA-Z0-9]* { yylval.name = strdup(yytext); return VARIABLE; }
";" { return SEPERATOR; }
[ \n\t]+ { }
. { return yytext[0]; }
%%
```

Execution Sequence:



```
PS E:\Semester 7\CC\Lab> bison -dy .\Practical_6_Yacc.y
PS E:\Semester 7\CC\Lab> flex .\Practical_6_Lex.l
PS E:\Semester 7\CC\Lab> gcc .\lex.yy.c .\y.tab.c -w
```

Output:



```
PS E:\Semester 7\CC\Lab> ./a.exe
a = p + (q - 10) / b * c;
temp0 = q - 10.000000
temp1 = temp0 / b
temp2 = temp1 * c
temp3 = p + temp2
a = temp3
Final Answer of Expression = temp3
```

Conclusion:

From this practical I learned how to Create 3AD (Three Address Code) from any given input expression using Lex and Yacc.