**Lab 6**

**Intermidiate code generator**

SRN:PES2UG23CS820

NAME:SANTHOSH A

**Lexer.l**

%{

#define YYSTYPE char\*

#include <stdio.h>

#include <string.h>

#include "y.tab.h" %}

digit [0-9] letter [a-zA-Z] id {letter}({letter}|{digit})\* digits {digit}+ opFraction (\.{digits})?

opExponent ([Ee][+-]?{digits})? number {digits}{opFraction}{opExponent}?

%option yylineno

%%

[ \t\r]+ ; // Skip whitespace \n { yylineno++; }

\/\/.\* ; // Skip comments

"=" { return '='; }

"(" { return '('; }

")" { return ')'; }

"" { return ''; }

"/" { return '/'; }

"+" { return '+'; } "-" { return '-'; }

{number} { yylval = strdup(yytext);

return T\_NUM;

}

{id} { yylval = strdup(yytext);

return T\_ID;

}

. ; // Ignore unknown characters

%%

int yywrap() {

return 1;

}

**Parser.y**

%{

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "quad\_generation.h" #define YYSTYPE char\*

extern int yylineno; int yylex(); void yyerror(const char\* s);

%}

%token T\_ID T\_NUM

%start START

%%

START:

ASSGN { printf("Valid syntax\n"); YYACCEPT; }

;

ASSGN:

T\_ID '=' E { quad\_code\_gen($1, $3, "=", "");

}

;

E:

E '+' T { char\* temp = new\_temp(); quad\_code\_gen(temp, $1, "+", $3);

$$ = temp;

}

| E '-' T { char\* temp = new\_temp();

quad\_code\_gen(temp, $1, "-", $3);

$$ = temp;

}

| T {

$$ = $1;

}

;

T:

T '\*' F { char\* temp = new\_temp(); quad\_code\_gen(temp, $1, "\*", $3);

$$ = temp;

}

| T '/' F { char\* temp = new\_temp(); quad\_code\_gen(temp, $1, "/", $3);

$$ = temp;

}

| F {

$$ = $1;

}

;

F:

'(' E ')' {

$$ = $2;

}

| T\_ID {

$$ = strdup($1);

}

| T\_NUM {

$$ = strdup($1);

}

;

%%

void yyerror(const char\* s) {

printf("Error: %s at line %d\n", s, yylineno);

}

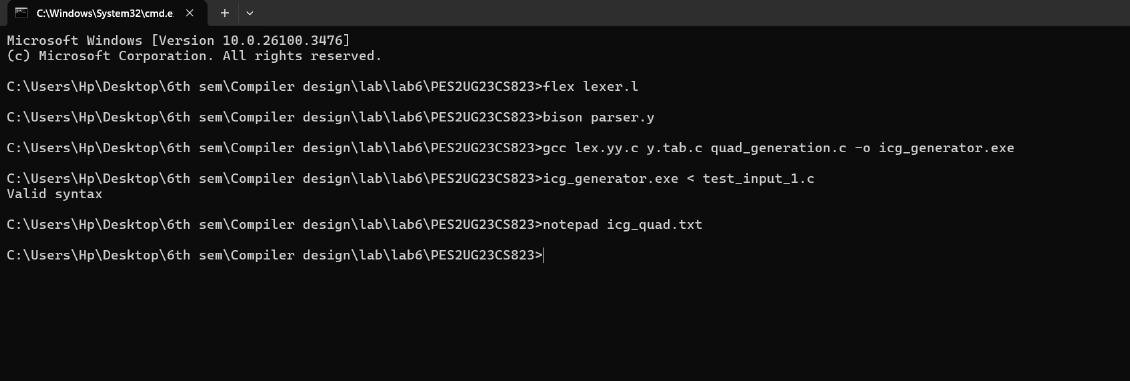
int main() { icg\_quad\_file = fopen("icg\_quad.txt", "w");

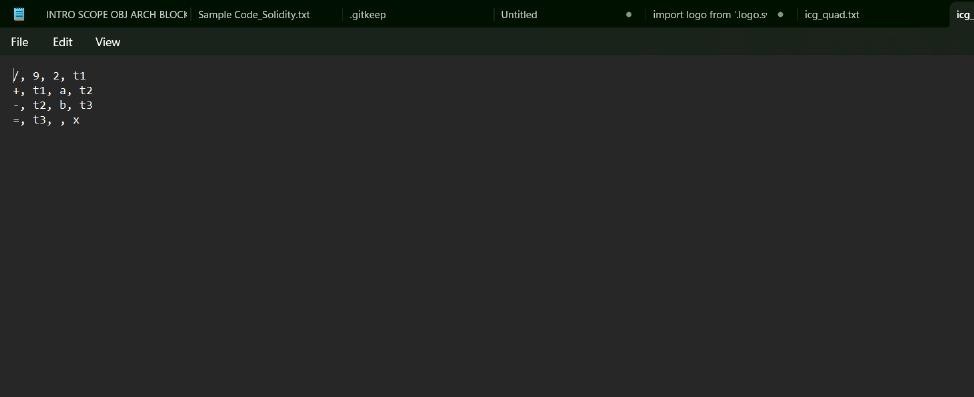
yyparse(); fclose(icg\_quad\_file);

return 0;

}

Test\_input\_1.c





Test\_input\_2.c

